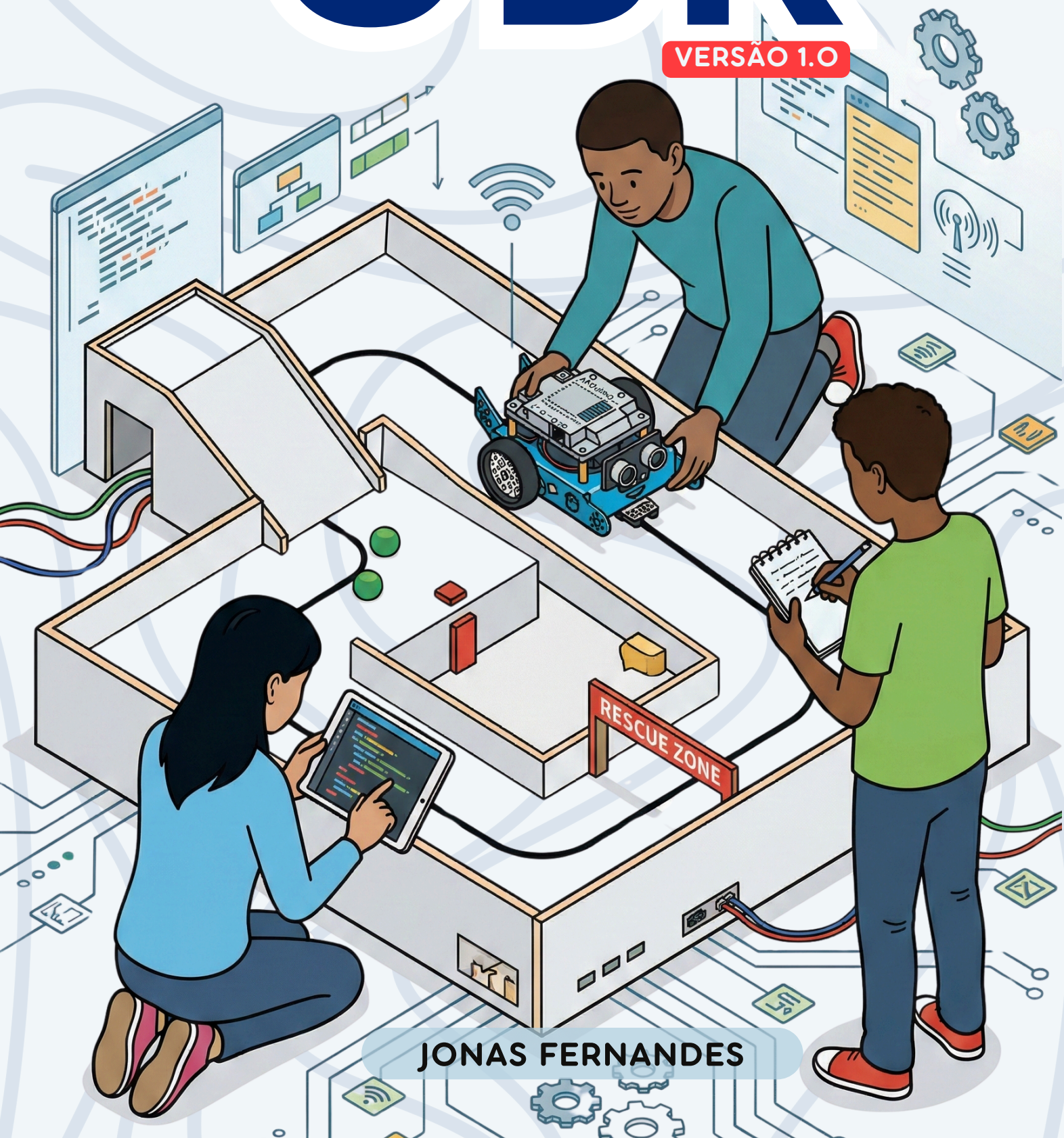
Guia Prático

OBR

VERSÃO 1.0



JONAS FERNANDES

Apresentação

Sabemos que a Robótica Educacional e competições como a Olimpíada Brasileira de Robótica (OBR) são ferramentas transformadoras. Elas impulsionam o desempenho nas áreas de ciência e tecnologia (STEM) e desenvolvem as habilidades essenciais do século XXI, os famosos 4Cs: Colaboração, Comunicação, Criatividade e Pensamento Crítico. Contudo, a adoção em larga escala dessas práticas, especialmente nas escolas públicas, esbarra em diferentes obstáculos, incluindo os de ordem financeira, formativa, além da dispersão de informações encontrada livremente na Internet, sem um roteiro sistematizado e específico para competições.

Foi para solucionar essa lacuna que desenvolvi este Guia Técnico-Pedagógico, um projeto fruto da minha especialização em Robótica Educacional. Ele nasceu da necessidade de criar uma ponte direta e sem rodeios entre a teoria vista na escola e a prática exigida na arena de competição.

Para garantir a máxima eficiência, estruturei o material fundamentado na abordagem Lean. Isso significa que eliminamos o desperdício teórico e focamos no que realmente gera valor: ensinar o estudante a pensar, projetar, programar e resolver problemas.

Acredito firmemente que a robótica vai muito além de montar e encaixar peças. Trata-se de desenvolver a capacidade de resolver problemas complexos com criatividade e trabalho em equipe.

Agradeço a todos pela atenção. Convido-os a mergulharem nas páginas deste guia para explorarmos, passo a passo, como preparar nossas equipes para os desafios da OBR.

Sumário

1 Apresentação do Guia

1.1 Propósito e escopo do guia 04
1.2 Público-alvo 05
1.3 Como este guia foi pensado 05
1.4 Como utilizar este guia 06

2 A Olimpíada Brasileira de Robótica

2.1 Panorama da Olimpíada Brasileira de Robótica 08
2.2 Modalidade prática: foco em resgate 09
2.3 Requisitos e regras essenciais 10
2.4 Pontuação e Sala de Resgate 11
2.6 Competências da equipe 13

3 Fundamentos de Robótica para a OBR

3.1 O que é um robô móvel autônomo 14
3.2 Seguidor de linha e tomada de decisão 16
3.3 Programação básica aplicada à prova da OBR 18

4 Trilha de Simulação com Tinkercad

4.1 Fundamentos da Simulação de Circuitos com Arduino 20
4.2 Prática: Montagem e Programação do Motor no Simulador 21
4.3 Expandindo as possibilidades de entradas e saídas 33

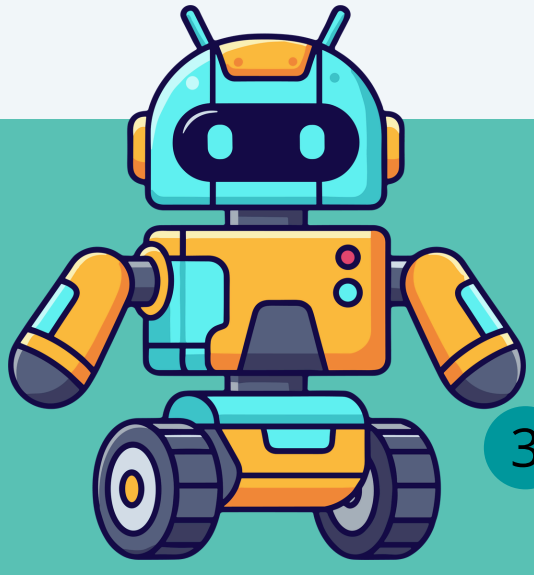
5 Do Virtual para o Real (A Montagem Física)

5.1 A Base Mecânica: Além de um Kit Comercial 37
5.2 A Ponte H e Baterias 38
5.3 Boas Práticas de Cabeamento 41
5.4 Posicionamento e Calibração do Sensor TCRT5000 43
5.5 O Teste de Bancada e o Primeiro Upload 45
5.6 Erros de iniciante e Resolução de Problemas 47

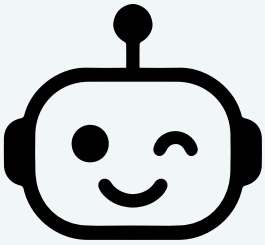
Conclusão

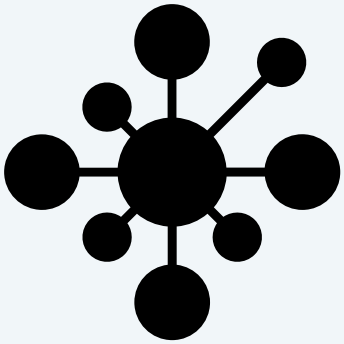
MODALIDADE PRÁTICA

GUIA OBR



Propósito e escopo do Guia 1.1





Este guia foi elaborado para apoiar escolas que desejam iniciar ou fortalecer a participação na modalidade prática da Olimpíada Brasileira de Robótica (OBR), com foco em alunos do 8º e 9º ano do Ensino Fundamental. Ele organiza uma trilha de aprendizagem que parte da simulação em ambiente virtual (Tinkercad) e evolui até a construção e programação de um robô físico compatível com a prova de **Rescue Line (Resgate de Linha)**, sempre de forma progressiva e realista quanto às limitações de tempo e recursos.

O escopo do material inclui:

- Conceitos fundamentais de robótica necessários para a OBR;
- Propostas de atividades práticas com uso do simulador Tinkercad;
- Simulação e projeto real com hardware Arduino para a montagem de um robô físico;
- Dicas de organização e montagem de componentes e testes de bancada.

IMPORTANTE

Este guia não tem como objetivo entregar um modelo único de robô pronto, mas sim fornecer referências, critérios e exemplos que permitam a cada equipe desenvolver sua própria solução, respeitando as regras da OBR e estimulando a criatividade dos estudantes.

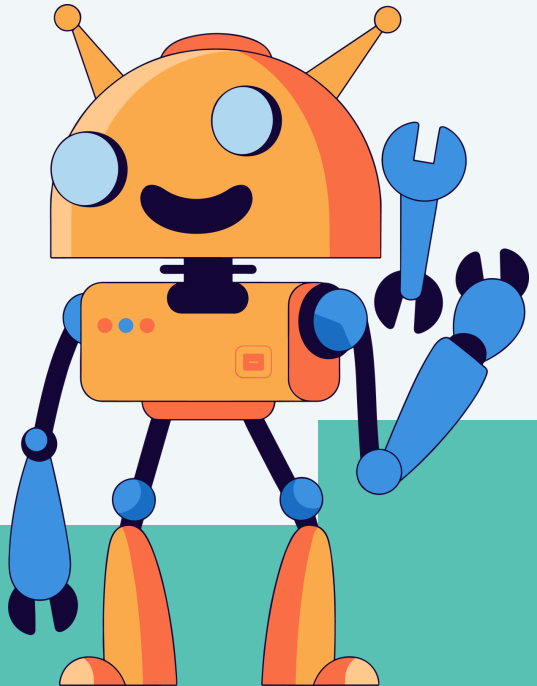
Público-alvo: 1.2

Professores e estudantes do 8° e 9° anos do Ensino fundamental

Professores: docentes de qualquer área com interesse em robótica educacional e na OBR, ainda que não tenham formação específica em eletrônica ou programação.
Estudantes do 8° e 9° ano: alunos iniciantes em robótica, com conhecimentos básicos de matemática e ciências e curiosidade para explorar tecnologia e desafios de lógica.

Como este guia foi pensado 1.3

Este guia foi projetado para que cada etapa tenha um propósito claro e aplicação direta na OBR Prática. Ao adotar a filosofia **Lean**, priorizou-se o que é essencial, eliminando excessos teóricos e focando em conteúdos que se transformam em ações concretas. O objetivo é reduzir o desperdício de tempo e garantir que cada lição aproxime alunos e professores do sucesso na competição.



A Robótica é dinâmica. Para acessar a versão mais atualizada deste guia, bem como arquivos extras de código e links de simuladores, acesse nossa pasta oficial e aberta no Google Drive:

Link GUIA Prático OBR

Como utilizar este guia

1.4

Este guia adotou uma abordagem ágil e direta (Lean), eliminando excessos teóricos para colocar o estudante rapidamente na prática. Ele foi pensado para ser flexível e pode ser utilizado de três formas principais:

O “Bootcamp” (Trilha Linerar)

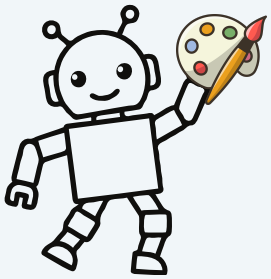
1

Ideal para equipes iniciantes. Os alunos seguem o guia do Capítulo 1 ao 5 na ordem, compreendendo as regras, validando toda a lógica de programação sem riscos no simulador Tinkercad e, logo em seguida, construindo o robô físico. É o caminho mais seguro para as primeiras semanas de preparação.

Manual de Bancada (Consulta Rápida):

2

Para equipes que já começaram a construir seus robôs, mas estão travadas. Os Capítulos 4 e 5 servem como guias de referência para problemas específicos: como ligar a Ponte H L298N sem queimar o Arduino, como calibrar os sensores para a luz da sala ou como distribuir a energia das baterias.



Guia de Troubleshooting (Resolução de Problemas):

3

O professor pode utilizar o guia (especialmente a seção 5.7 de Erros de Iniciante) como ferramenta de autonomia. Quando o robô físico apresentar falhas, em vez de dar a resposta, o educador orienta os alunos a consultarem o manual para investigarem se o problema é de hardware (fios invertidos, mau contato) ou software.

IMPORTANTE

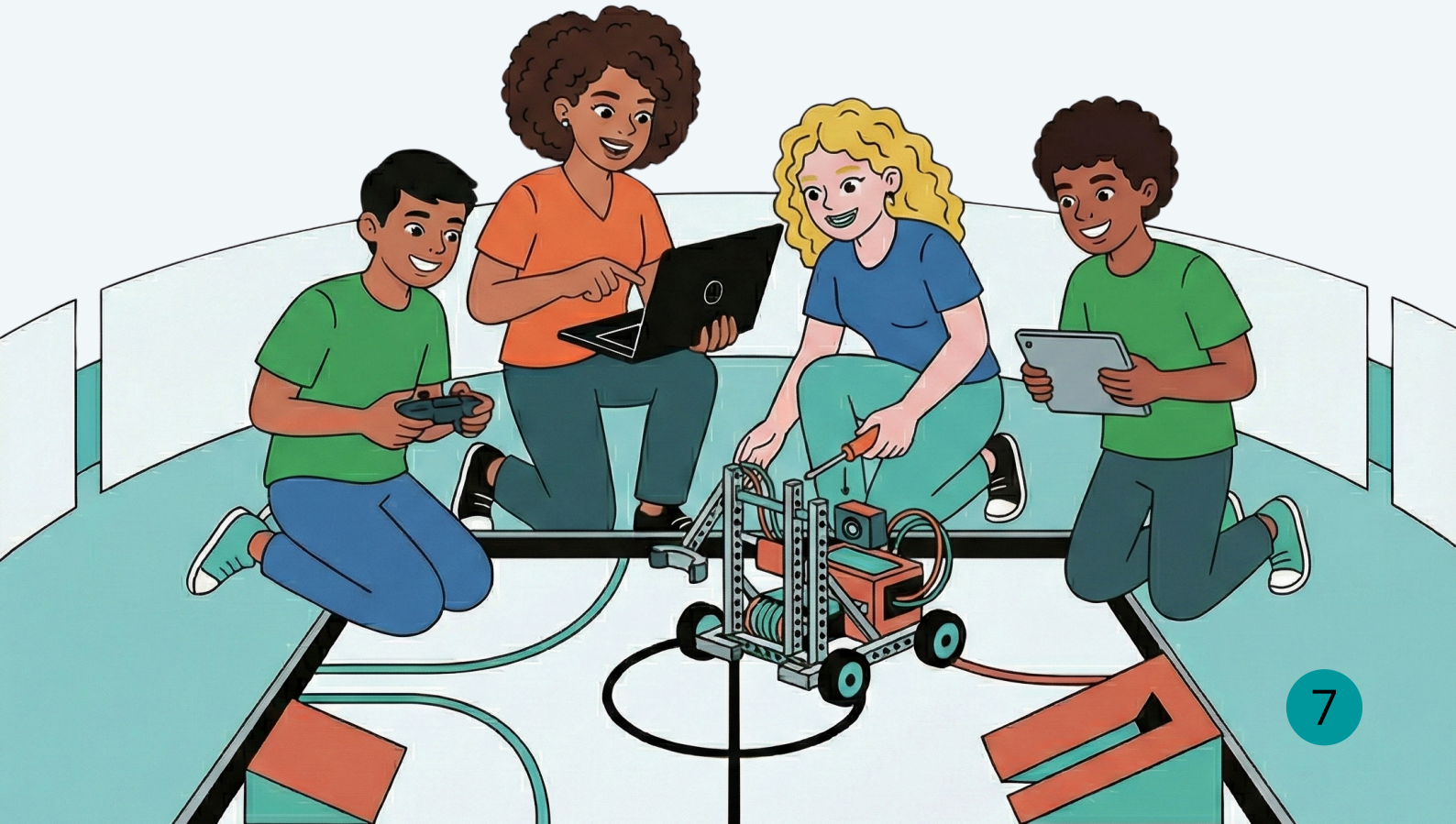
O segredo para uma preparação eficiente é a flexibilidade: escolha a trilha de uso que melhor se encaixe no seu calendário, nos recursos da bancada e no nível técnico dos alunos.

A Olímpiada Brasileira de Robótica

2

MISSÃO OBR

Promover, incentivar e disseminar a robótica pelo Brasil

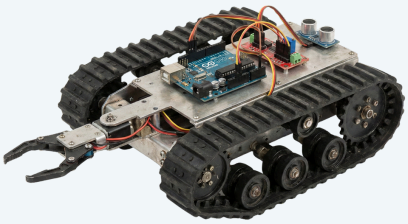


Panorama da Olimpíada Brasileira de Robótica

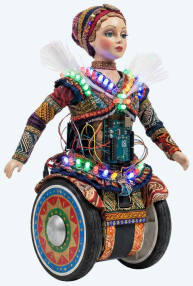
2.1

A Olimpíada Brasileira de Robótica (OBR) é uma competição educacional que busca aproximar estudantes, de 6 a 19 anos, da ciência, tecnologia e engenharia por meio de desafios de robótica. Ela se divide nas modalidades prática e teórica com diferentes níveis de experiência, sempre com foco em resolução de problemas e trabalho em equipe.

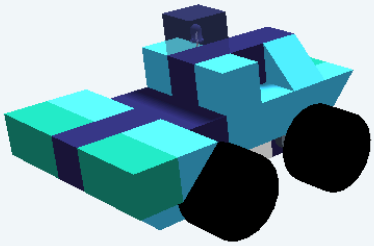
Na modalidade **prática**, os estudantes constroem e programam robôs autônomos para executar tarefas em arenas que simulam situações do mundo real. A modalidade prática é dividida nas categorias Robótica de **Resgate** e **Artística**, que ocorrem de forma presencial; e a Robótica **Virtual** com a possibilidade de competir através de simuladores, promovendo a inclusão de escolas com dificuldades de deslocamento nas etapas regionais.



Robótica de Resgate



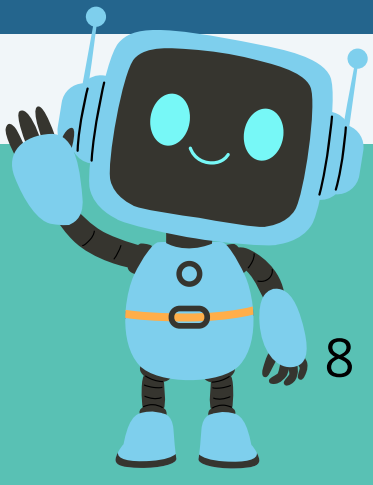
Robótica Artística



Robótica Virtual

Mais informações sobre a **modalidade prática** estão disponíveis em::

https://obr.robocup.org.br/modalidade-pratica/



Modalidade Prática: foco em resgate

2.2

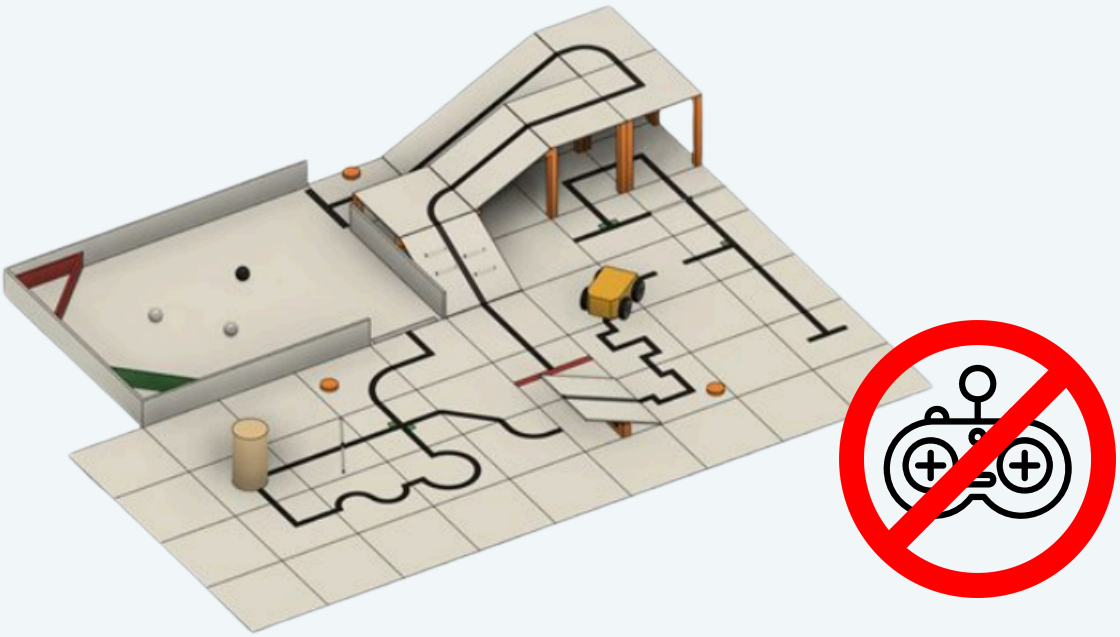
Dentro da modalidade prática, a robótica de **resgate em linha** tem como objetivo simular um resgate de vítimias em situação de desastre utilizando um robô completamente **autônomo** e sem qualquer assistência humana.

Meta OBR

Realizar o resgate de vítimas sem a necessidade de interferência humana.

O robô deve **seguir uma linha** escura em um piso claro (Figura 2.1) e superar obstáculos de diferentes dificuldades, desviar de objetos desconhecidos, superar caminhos sem saída e enfrentar outros contratempos para conseguir **transportar as vítimas** para uma região segura. Após o resgate, o robô segue a missão até ultrapassar a linha de chegada.

Figura 2.1 - Arena modelo da OBR 2025



Fonte: Manual da OBR 2025

A arena simula um ambiente de desastre, em que a linha representa o caminho seguro até as vítimas. O robô deve ser **totalmente autônomo**, ou seja, sem controle remoto e deve tomar decisões a partir das leituras de sensores de linha e outros sensore para desviar de obstáculos. Isso faz com que o desafio combine aspectos de eletrônica, programação, controle de movimento e estratégia de navegação.

Requisitos e regras essenciais

2.3

Embora o regulamento da OBR seja atualizado periodicamente e deva sempre ser consultado na versão vigente, alguns elementos se mantêm relativamente estáveis e são fundamentais para orientar o planejamento de professores e alunos. Entre eles, destacam-se:

Especificações do robô:

O que deve ser feito:

Um robô autônomo;

Montagem, desenvolvimento e programação de robôs apenas por estudantes;

Iniciar o desafio manualmente pelo capitação da equipe;

Utilizar quantos sensores forem necessários;

Construção de alça para transporte do robô móvel e botão de início da rodada;

Documentação da trajetória do projeto com registro de hardware e software do robô;

O que não pode:

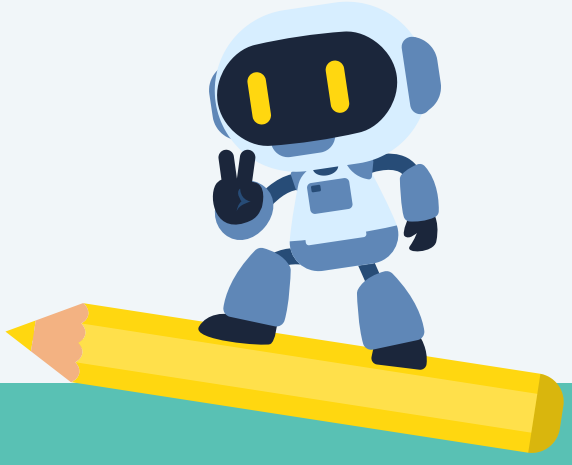
Usar um robô com possibilidade de comunicação sem fio para controle remoto;

Realizar pré-mapeamento da arena no dia da competição e progrmar o robô para realizar os movimentos pré-definidos.

Usar kits comercialmente da categoria “seguidor de linha” ou “robô para resgate”;

Uso de câmers com pré-programação para identificação de objetos;

Compartilhar robôs entre equipes durante a competição;



Pontuação e Sala de resgate

2.4

Atribuição de pontos é feita de acordo com o progresso do robô na arena, a conclusão de trechos, o sucesso nas ações de resgate e o cumprimento de critérios definidos no manual. A versão do manual da OBR 2025 apresenta a pontuação na Figura 2.2.

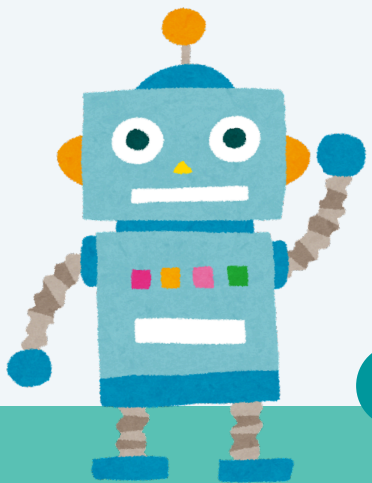
Figura 2.2 - Tabela de pontuação na OBR 2025

Superar Interseção ou Beco sem Saída	**10 Pontos**
Superar uma Gangorra	**20 Pontos**
Desviar de um Obstáculo	**20 Pontos**
Superar um Ladrilho Contendo Gap(s)	**10 Pontos**
Subir ou Descer cada ladrilho contendo uma rampa	**10 Pontos**
Superar um ladrilho contendo Lombada(s)	**10 Pontos**
Superar uma Passagem	**0 Pontos**
Ladrilho de Chegada	**60 Pontos**

Fonte: Manual da OBR 2025

Fique esperto!

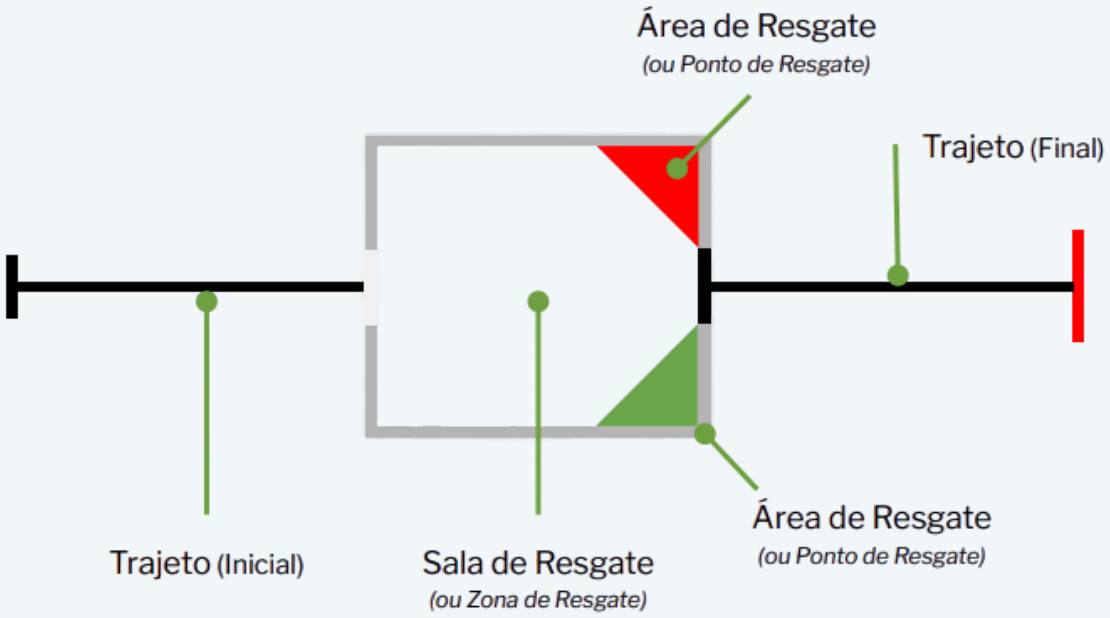
Existe Limitação de tempo de prova, o que exige que o robô seja não apenas funcional, mas também eficiente no percurso.



A Sala de resgate

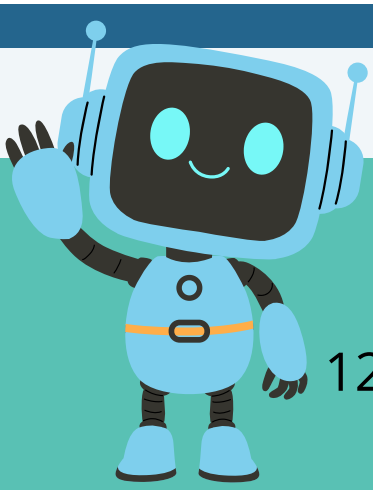
A sala de resgate é um espaço quadrado ou retangular na arena (Figura 2.3) com entrada demarcada por um fita no piso de cor prata e a saída por uma fita de cor preta. Nesta sala, o robô deve localizar e transportar vítimas vivas (esferas de cor prata) para a área de resgate verde. Vítimas mortas (esferas escuras) devem ser transportadas para a área vermelha. Após o resgate, o robô deve sair da sala de resgate e seguir a linha até o ladrilho de chegada representado por uma linha vermelha.

Figura 2.3 - Espaço Sala de Resgate na arena da OBR



Fonte: Manual da OBR 2025

Manual completo de construção da arena da OBR pode ser acessado em:
https://obr.robocup.org.br/documentos-e-manuais/



Competências da equipe

2.6

A participação na OBR prática, especialmente no desafio de **Resgate**, envolve um conjunto de competências técnicas e socioemocionais que vão além da simples montagem de um robô. Este guia organiza essas competências em quatro grandes eixos:

Técnicas de Robótica

- Principais componentes (sensores de linha, motores, controlador, fonte de energia).
- Saber montar circuitos básicos com segurança e interpretar esquemas simples.
- Desenvolver e ajustar algoritmos de seguidor de linha e tomada de decisão em situações como curvas, cruzamentos e obstáculos.

Técnicas de Programação

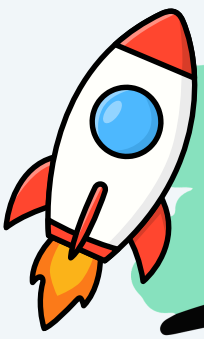
- Estruturar programas com início, configuração e laço principal, usando variáveis de forma adequada.
- Utilizar comandos de repetição e condição para reagir às leituras dos sensores.
- Depurar o código por meio de testes sistemáticos, observando o comportamento do robô e registrando erros e melhorias.

Trabalho em Equipe

Distribuir papéis dentro da equipe (por exemplo, responsável por programação, por montagem, por testes e registros).
Planejar o trabalho ao longo do tempo, definindo metas de curto prazo.
Comunicar resultados, dificuldades e soluções, mantendo um caderno de bordo atualizado.

Habilidades Socioemocionais

Lidar com frustrações, uma vez que erros, falhas de montagem e comportamentos inesperados do robô fazem parte do processo.
Desenvolver perseverança, revisitando soluções e tentando novas abordagens quando algo não funciona.
Exercitar a capacidade de escuta e de negociação de ideias dentro da equipe, respeitando diferentes pontos de vista.



O sucesso não depende apenas de "saber programar", mas de um conjunto integrado de habilidades desenvolvidas ao longo das trilhas deste material.

Fundamentos de Robótica para a OBR

3

Este capítulo apresenta os conceitos mínimos de robótica e programação necessários para que professores e estudantes compreendam o funcionamento de um robô seguidor de linha voltado para a OBR prática. A ideia não é aprofundar toda a teoria, mas oferecer uma base clara, conectada diretamente aos desafios de Rescue Line.

3.1 O que é um robô autônomo?

Um robô móvel autônomo é uma máquina capaz de se deslocar pelo ambiente tomando decisões por conta própria, com base em informações captadas por sensores. Diferente de um brinquedo de controle remoto, ele não depende de comandos diretos do competidor durante a prova; o comportamento é definido previamente por meio de programação.

Direto ao ponto

Esqueça o controle remoto. Na OBR, você não pilota o robô; você ensina o robô a se pilotar.

O ciclo SPA

Sentir - Processar - Agir

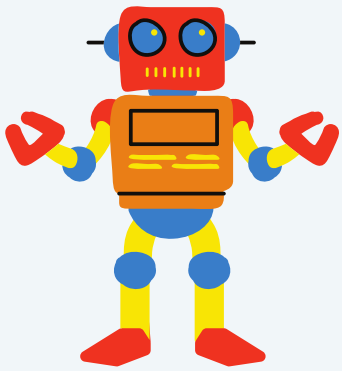
Para entender como ele funciona, imagine um "aluno que enxerga o chão, pensa rapidamente e decide para onde virar". O robô imita exatamente esse comportamento humano através do ciclo SPA **(Sentir - Processar - Agir)**:

- **Sentir (Sensores):** Assim como seus olhos veem a linha preta no chão, os Sensores de Refletância "enxergam" o contraste entre o claro e o escuro. Em alguns projetos, podem ser usados sensores de distância para detectar obstáculos.

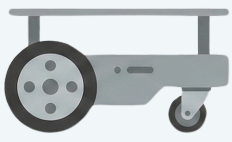
- **Processar (Controlador):** Assim como seu cérebro decide "fazer uma curva fechada", a placa Arduino lê os dados dos sensores e processa o código seguindo um programa previamente escrito e decide como acionar os motores. É nessa etapa que entram as estruturas de decisão (se... então...) e os algoritmos de seguidor de linha.

- **Agir (Atuadores):** Os atuadores são os componentes que realizam ações físicas. No robô da OBR, os principais atuadores são os motores responsáveis por girar as rodas, permitindo ao robô andar para frente, para trás ou virar. Em alguns casos, podem existir outros atuadores (como servomotores para mecanismos de resgate), dependendo da estratégia adotada pela equipe.

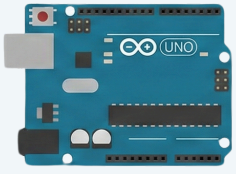
O robô autônomo é uma máquina que toma decisões sozinha baseada no que "vê" (sensores) e no que "sabe" (seu código).



Para a OBR prática, o robô precisa ter:



Um chassi com rodas (plataforma)



Placa controladora (Arduíno)

Sensores

Motores

Seguidor de linha e tomada de decisão

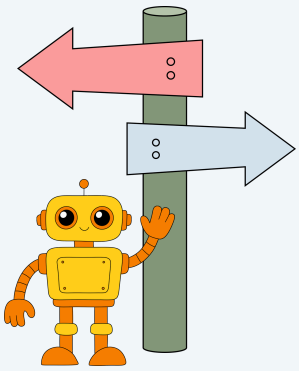
3.2

O desafio na robótica de resgate se baseia na capacidade do robô de seguir uma trilha desenhada no chão, mesmo em situações em que a linha apresenta curvas ou interrupções. Para isso, o robô precisa medir constantemente o contraste entre linha e fundo por meio do sensor reflexivo, decidir em que direção ajustar o movimento e repetir esse processo rapidamente.

Alguns conceitos centrais:

- **Linha e fundo:** em geral, a linha é escura e o fundo é claro (ou o contrário), o que permite ao sensor distinguir entre “em cima da linha” e “fora da linha”.

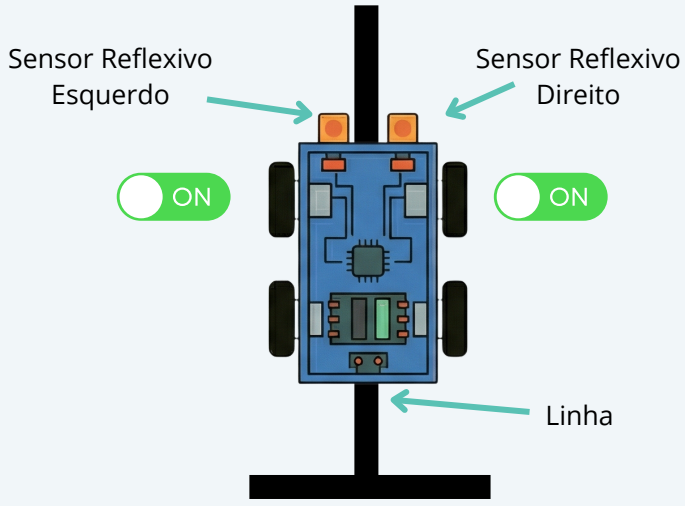
- **Posição relativa:** com dois ou mais sensores de linha, o robô consegue identificar se está centralizado, deslocado para um lado ou completamente fora da trilha.



REGRAS DE DECISÃO

O comportamento do robô é definido por regras definidas na programação da placa controladora. Em geral, essas regras dependem de cada situação “sentida” através dos sensores.

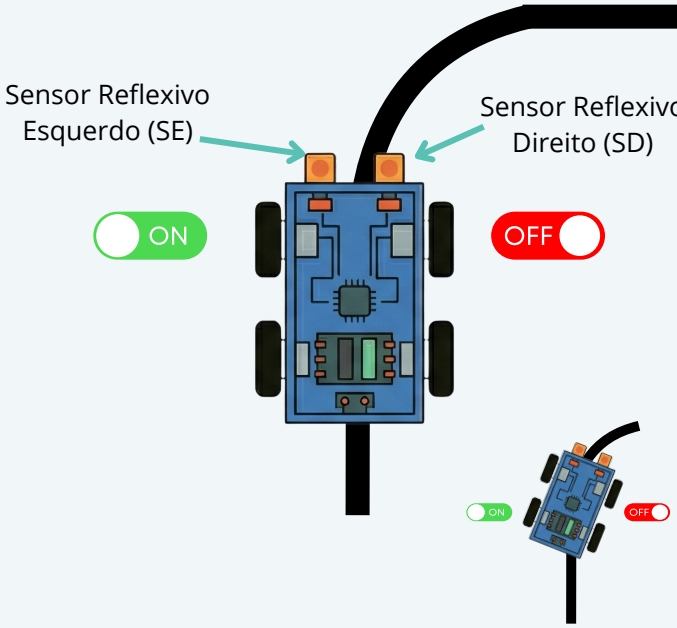
Situação 1: SIGA EM FRENTE



SIGA EM FRENTE

Se os sensores DIREITO e ESQUERDO recebem luz (reflexão devida a cor branca), ambos motores permanecem ligados e o robô deve seguir em frente.

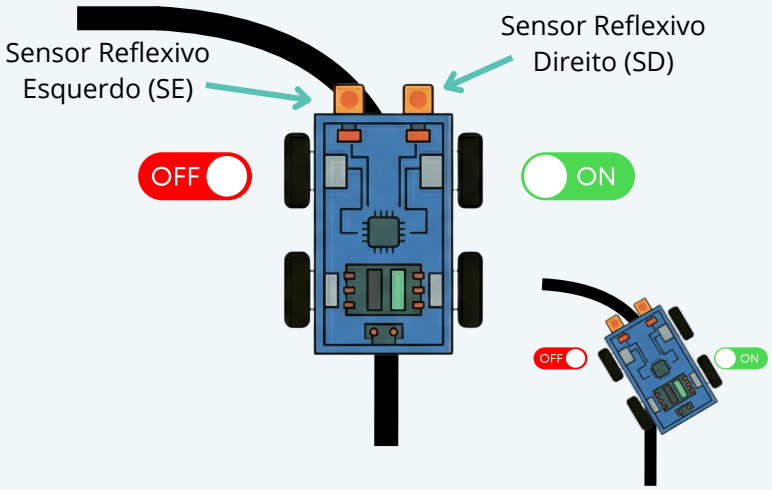
Situação 2: CORRIJA PARA A DIREITA



Curva à Direita

Se o sensor DIREITO parar de receber luz (ver preto) e o ESQUERDO continuar vendo branco: O robô deve desligar o motor da roda direita e ligar a esquerda para fazer uma curva à direita.

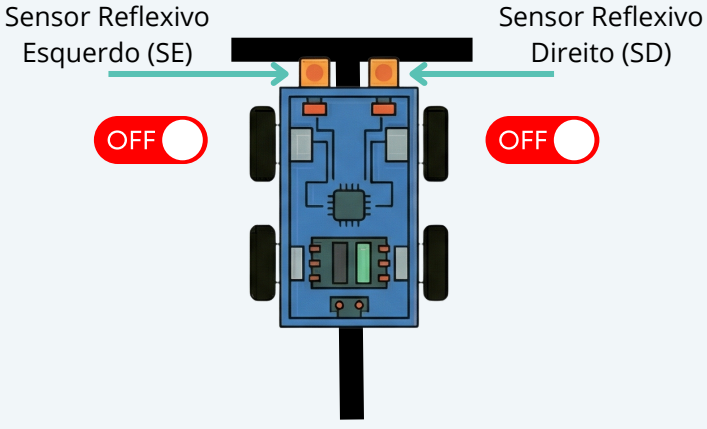
Situação 3: CORRIJA PARA A ESQUERDA



Curva à Esquerda

Se o sensor ESQUERDO parar de receber luz (ver preto) e o DIREITO continuar vendo branco: O robô deve frear a roda esquerda e acelerar a direita para voltar ao centro.

Situação 4: CRUZAMENTO (DECISÃO)



PARE

Se AMBOS os sensores verem preto: O robô identificou um cruzamento ou o final do percurso. A programação deve decidir se ele para ou segue em frente (contagem de tempo ou verificação de outros sensores).

Programação básica aplicada à prova da OBR

3.3

Na OBR, um robô bem montado sem um bom código é apenas uma escultura de metal e plástico. A programação é o que transforma decisões lógicas em movimento físico. Para dominar a arena, você precisa entender que o código não é mágica; é uma receita que o robô segue cegamente. Vamos dividir essa "receita" em 4 pilares essenciais:

1. A Estrutura: Acordar e Correr (Setup & Loop)

Todo código de Arduino (e da maioria dos robôs) é dividido em dois momentos distintos:

Configuração

setup

Ocorre apenas uma vez quando o robô é ligado.
O que fazer aqui: Definir quem são as entradas (sensores) e as saídas (motores).

Ciclo Principal

loop

Ocorre continuamente e infinitamente.
O que fazer aqui: Ler Sensores → Tomar Decisão → Mover Motores.

2. Comandos de Decisão: O Poder do "SE"

O robô não adivinha; ele compara. A principal ferramenta lógica na OBR é a estrutura Condicional (if... else).

Comando

Se (algo acontecer)...
Então (faça isso)...
Senão (faça aquilo)

Direto ao ponto

SE (Sensor Esquerdo ver PRETO) →
ENTÃO (Vire para a Esquerda).
SENÃO → (Siga em frente).

Ao final deste capítulo, espera-se que professores e alunos tenham uma visão funcional de como um robô da OBR **"percebe, pensa e age"**, compreendendo que a programação é um instrumento para transformar decisões lógicas em comportamentos concretos na arena. Isso prepara o terreno para as trilhas de simulação em **Tinkercad** e **SBotics**, nas quais esses conceitos serão experimentados na prática.

Trilha de simulação com Tinkercad

4

Este capítulo introduz o simulador Tinkercad como um laboratório virtual seguro para os primeiros passos na montagem do robô. O objetivo não é formar especialistas em eletrônica, mas permitir que alunos e professores validem circuitos, testem motores e sensores de forma visual antes de manipularem os componentes físicos, eliminando o risco de danos às peças reais e acelerando a preparação para a OBR.

Fundamentos da Simulação de Circuitos com Arduino

4.1

A simulação de circuitos é uma etapa fundamental no desenvolvimento de projetos de eletrônica e robótica, especialmente em ambientes educacionais. Antes de manipularmos os componentes físicos, o ambiente virtual nos permite **testar** ideias, **validar** a lógica de programação e **corrigir** erros de fiação sem o risco de queimar componentes reais, como o próprio Arduino, motores ou sensores.

Vantagens da simulação

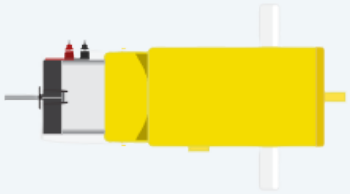
Todo código de Arduino (e da maioria dos robôs) é dividido em dois momentos distintos:

Segurança e Economia: Erros de polaridade ou curtos-circuitos no simulador resultam apenas em um aviso na tela, preservando equipamentos que custariam caro no mundo real.

Acessibilidade: Permite que os estudantes comecem a programar e montar circuitos mesmo antes de terem o hardware físico em mãos ou quando estão estudando em casa.

Análise Visual: Plataformas de simulação geralmente oferecem ferramentas visuais, como multímetros e osciloscópios virtuais, que ajudam a entender o fluxo de corrente e os sinais elétricos em tempo real.

O motor DC na simulação



Neste capítulo, focaremos no acionamento de um Motor de Corrente Contínua (DC). Em um simulador, o motor responde aos comandos do Arduino de forma idêntica à realidade: girando em velocidades variadas ou alterando seu sentido de rotação com base no código programado. É importante notar que, assim como na vida real, um motor exige mais corrente do que as portas digitais do Arduino podem fornecer diretamente. Por isso, a simulação também envolverá o uso de componentes de interface, como transistores (ex: TIP120) ou Pontes H (ex: L293D).

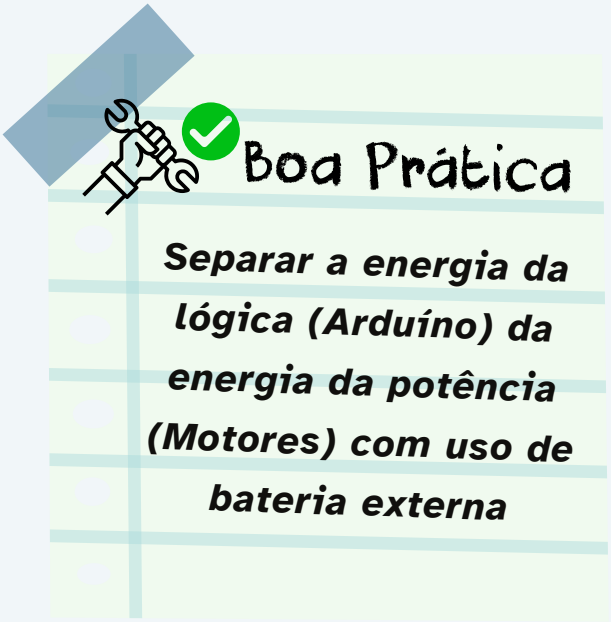
Prática: Montagem e Programação do Motor no Simulador

4.2

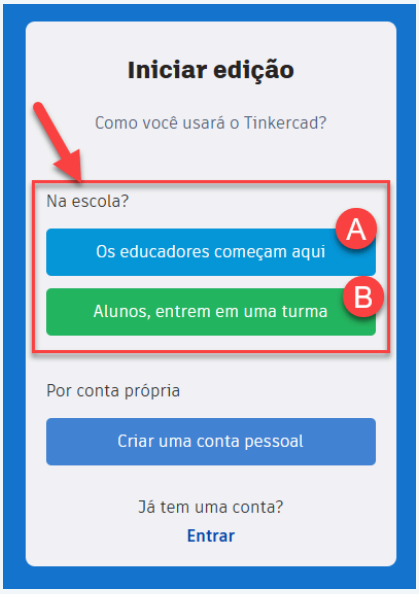
Agora que entendemos a importância do ambiente virtual, vamos construir nosso primeiro circuito interativo. O objetivo desta prática é fazer o simulador girar um Motor DC controlando sua ativação através do código do Arduino.

Componentes Virtuais Necessários:

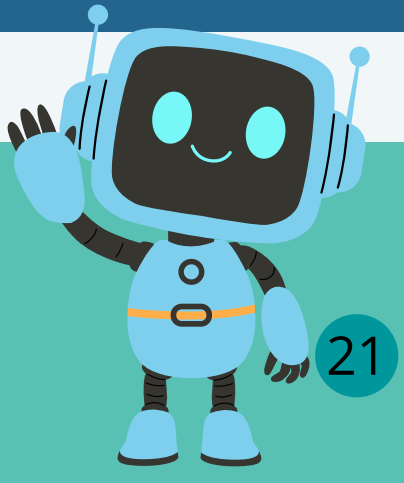
1x Placa Arduino Uno R3
1x Protoboard (Placa de ensaio)
1x Motor DC
1x Circuito Integrado Ponte H (L293D)
1x Bateria de 9V (alimentação dedicada do motor)



Crie sua conta gratuita no botão **Inscrever-se** no site do Tinkercad e escolha a opção “**educador**” ou “**aluno**”, de acordo com o seu perfil



O simulador Tinkercad pode ser acessado em:
https://www.tinkercad.com/



MISSÃO 1

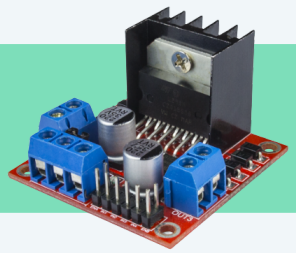
Dando vida aos motores (Controle de direção)

Para dar verdadeira mobilidade a um robô, especialmente ao prepará-lo para os desafios de competições como a Olimpíada Brasileira de Robótica (OBR), o controle bidirecional é essencial. Precisamos que o robô seja capaz de avançar, recuar e fazer curvas precisas. É aqui que entra a Ponte H (L293D).

Este circuito integrado é o nosso "**cérebro de potência**". Ele substitui montagens complexas com componentes soltos, permitindo inverter a polaridade da corrente (fazendo o motor girar em ambos os sentidos) e já possuindo proteção interna para evitar danos à placa Arduino.

Direto ao ponto

Na vida real o componente L293D será substuído pela ponte H L298N



Posicionamento: Coloque o chip L293D bem no centro da protoboard, dividindo as duas metades da placa.

Alimentação da Lógica e Potência:

- Conecte o pino 5V do Arduino à linha de energia positiva superior da protoboard e o GND à linha negativa superior.
- Conecte o positivo da Bateria de 9V à linha de energia positiva inferior, e o negativo da bateria à linha negativa inferior.

Atenção: Una as duas linhas negativas (GND do Arduino e GND da bateria) com um fio para criar uma referência comum de terra. Sem isso, o circuito não funciona.

Conexões de Energia do L293D (Lado esquerdo do chip):

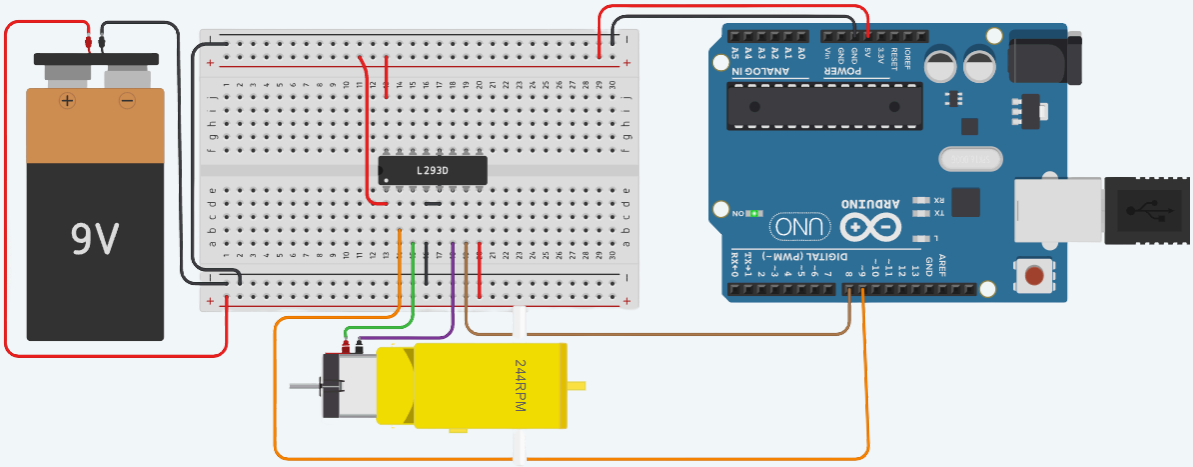
- Pino 1 (Ativar 1 e 2): Conecte aos 5V da protoboard (isso mantém o motor sempre ativado na velocidade máxima).
- Pino 16 (Potência Lógica): Conecte aos 5V do Arduino.
- Pino 8 (Potência do Motor): Conecte aos 9V da bateria.
- Pinos 4 e 5 (Terra/GND): Conecte à linha negativa comum.

Conexão do Motor e Controle:

Para que a Ponte H consiga inverter o sentido do giro, ela precisa controlar a entrada e a saída da corrente elétrica. Portanto, usaremos duas saídas diferentes do chip conectadas aos dois bornes distintos do mesmo motor, conforme a Figura 1 e os passos abaixo:

- Conecte o terminal vermelho do Motor DC ao Pino 3 (Saída 1) do L293D.
- Conecte o terminal preto do mesmo Motor DC ao Pino 6 (Saída 2) do L293D. (Nunca ligue os dois pinos do chip no mesmo terminal do motor, pois a corrente precisa entrar por um fio e sair pelo outro para gerar movimento).
- Por fim, conecte os Pinos 2 e 7 (Entrada 1 e Entrada 2) do L293D aos pinos digitais 9 e 8 do Arduino, respectivamente. O Arduino enviará os comandos por estes fios para avisar à Ponte H para qual lado a corrente deve fluir.

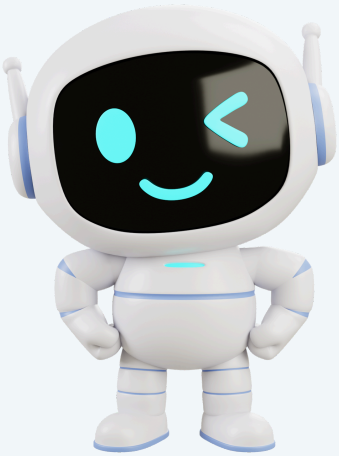
Figura 4.1 - Circuito de funcionamento do motor no Tinkercad



Fonte: Autoria própria

A Lógica de Programação (Código C++):

A lógica de **programação em C++** atua como o cérebro da nossa simulação, traduzindo nossas intenções em comandos elétricos que o Arduino consegue compreender e executar. Na prática, escrevemos um conjunto sequencial de instruções utilizando funções nativas da linguagem para manipular as portas do microcontrolador.



Saiba mais sobre Programação com Arduino:

Vídeo: Programando com Arduíno no Tinkercad

YouTube

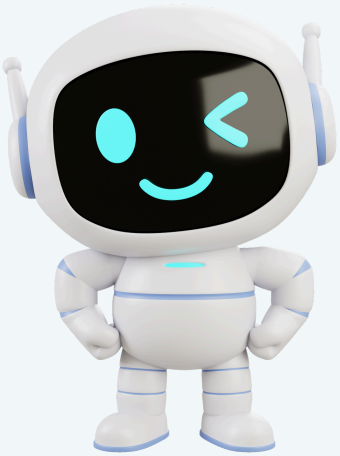
O Código:

Ao estruturarmos o código nas áreas fundamentais **setup()** e **loop()**, conforme a Figura 2, definimos a configuração inicial dos pinos e, em seguida, criamos um ciclo contínuo de envio de sinais de nível ALTO **(HIGH)** ou BAIXO **(LOW)**; são exatamente essas alternâncias lógicas, combinadas com intervalos de tempo (como o comando delay), que comandam a Ponte H para acionar o motor, inverter a sua rotação ou freá-lo, transformando linhas de texto em movimento real para o robô. Veja na Figura 4.2 uma ilstração do código em C++.

Figura 4.2 - Exemplo de código C++ para acionamento de motor

```
  //Definir pinos de controle da Ponte H
const int pinoFrente = 9;
const int pinoTras = 8;

void setup(){
  //Configura os pinos como saída
 pinMode (pinoFrente, OUTPUT);
 pinMode (pinoTras, OUTPUT);
 Serial.begin(9600);
}

void loop(){
  //Rodar o motor para FRENTE
  //Um pino em nível ALTO e o outro em nível BAIXO
    digitalWrite(pinoFrente, HIGH);
    digitalWrite(pinoTras, LOW);
    delay(2000); //mantém movimento por dois segundos

  //Para os motores
    digitalWrite(pinoFrente, LOW);
    digitalWrite(pinoTras, LOW);
    delay(2000); //Fica parado por dois segundos

  //Rodar o motor para TRÁS
    digitalWrite(pinoFrente, LOW);
    digitalWrite(pinoTras, HIGH);
    delay(2000); //matmém movimento por dois segundos

  //Para os motores
    digitalWrite(pinoFrente, LOW);
    digitalWrite(pinoTras, LOW);
    delay(2000); //Fica parado por dois segundos
}
```

Fonte: Autoria própria



Saiba mais sobre Programação no Arduino:

Vídeo: Programando no Arduino com C++

YouTube

MISSÃO 2

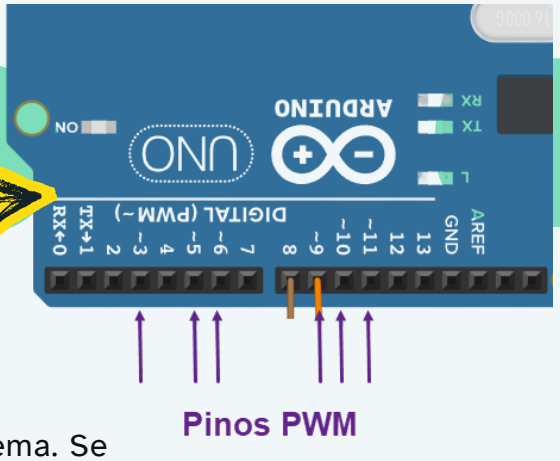
Dominando a Velocidade com PWM

Na "Missão 1", o nosso robô aprendeu a ir para frente e para trás, mas ele estava operando sempre com "*o pé na tábua*", ou seja, em sua potência máxima. Em uma arena de resgate da OBR, um robô que se move rápido demais tem grande chance de passar direto por um cruzamento, perder a fita no modo seguidor de linha ou até mesmo tombar.

Para termos controle total da tração, precisamos de um **acelerador**. No Arduino, não conseguimos simplesmente mandar "meia energia" (como 2.5 Volts) para o motor. A placa só entende ligado (5V) ou desligado (0V). A mágica acontece usando uma técnica chamada **PWM** (*Pulse Width Modulation*, ou Modulação por Largura de Pulso).

Direto ao ponto

No Arduino, os pinos PWM são marcados com um til (~) ao lado do número.



Como o PWM funciona na prática?

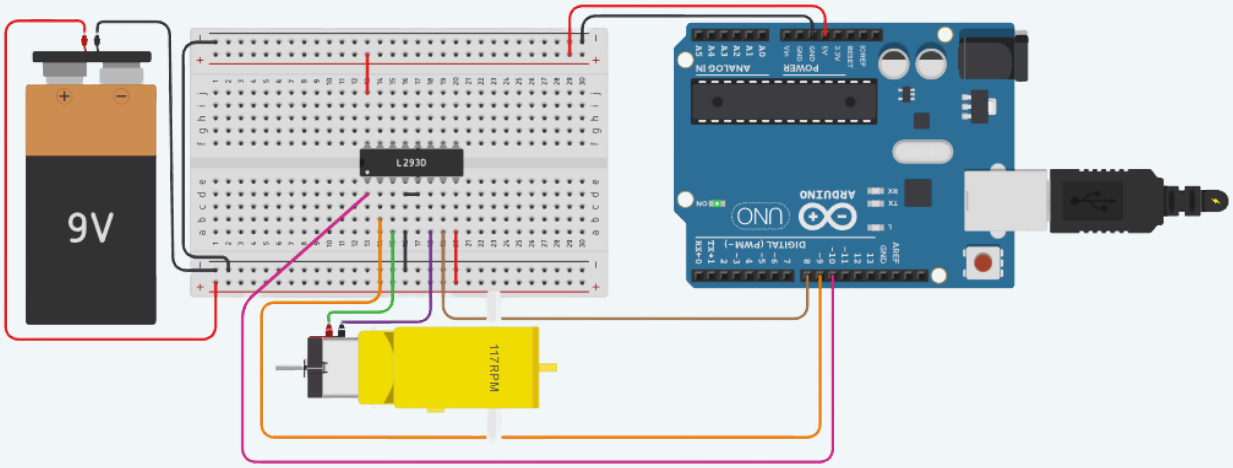
O Arduino liga e desliga o pino numa velocidade extrema. Se ele passa 50% do tempo ligado e 50% desligado, o motor não tem tempo de parar e "sente" como se estivesse recebendo metade da energia, girando mais devagar. Quanto mais tempo o pulso fica em nível ALTO (Duty Cycle), mais rápido o motor gira.

Ajuste na Montagem Virtual:

Para controlar a velocidade usando a Ponte H L293D, não vamos mexer nos fios de direção (pinos 2 e 7 do chip). Vamos assumir o controle do pino de **Enable** (Ativar) através da conexão PWM com o pino 10 (fio rosa).

1. Volte ao seu circuito da Missão 1 no simulador.
2. Lembra do Pino 1 (Ativar 1 e 2) do L293D, que havíamos ligado direto nos 5V da protoboard? Desconecte esse fio.
3. Conecte este Pino 1 a um pino do Arduino que suporte PWM (Pino 10, conforme a Figura 4.3).

Figura 4.3 - Circuito com velocidade definida por PWM



Fonte: Autoria própria

A Lógica de Programação (Código C++):

No código da Figura 4.4, usaremos a função **analogWrite()**. Ela atua no pino **PWM** e recebe valores numéricos que variam de **0** (totalmente parado) até **255** (velocidade máxima).

O Código:

Ao analisar o código, uma dúvida muito comum pode surgir: por que precisamos manter as linhas **digitalWrite(pinoFrente, HIGH)** e **digitalWrite(pinoTras, LOW)** se agora estamos controlando o motor com o analogWrite? A resposta está na divisão de tarefas da Ponte H. Os comandos ***digitalWrite*** funcionam como a "caixa de marcha" do nosso robô, definindo exclusivamente o sentido em que a corrente elétrica vai fluir (neste caso, para frente). Já o novo comando **analogWrite** atua puramente como o **"pedal do acelerador"** no pino de ativação, ditando a força com que essa corrente passa.

Figura 4.4 - Exemplo de código C++ para acionamento com PWM

```
  //Definir pinos de controle da Ponte H
const int pinoFrente = 9;
const int pinoTras = 8;
const int pinoVelocidade = 10; //Controle de velocidade PWM

void setup(){
  //Configura os pinos como saída
 pinMode (pinoFrente, OUTPUT);
 pinMode (pinoTras, OUTPUT);
 pinMode (pinoVelocidade, OUTPUT);
}

void loop(){
  //Rodar o motor para FRENTE
  //Um pino em nível ALTO e o outro em nível BAIXO
    digitalWrite(pinoFrente, HIGH);
    digitalWrite(pinoTras, LOW);

  //Acelera o motor em velocidade máxima
    analogWrite(pinoVelocidade, 255);
    delay(2000); //mantém movimento por dois segundos

  //Reduz velocidade para metade
    analogWrite(pinoVelocidade, 127);
    delay(2000); //mantém movimento por dois segundos

  //Reduz para velocidade 25%
    analogWrite(pinoVelocidade, 64);
    delay(2000); //mantém movimento por dois segundos
}
```

Fonte: Autoria própria

MISSÃO 3

Tração Dupla (Controlando dois Motores)

Até agora, usamos apenas um lado da nossa Ponte H L293D. A grande vantagem deste chip é que ele é duplo. Ele foi projetado exatamente para controlar dois motores simultaneamente, de forma totalmente independente.
Para que um robô faça uma curva, nós usamos a lógica estudada na seção 3.2 deste guia:

- **Para ir reto:** Os dois motores giram para frente na mesma velocidade.
- **Para virar à direita:** O motor esquerdo gira para frente e o motor direito para (ou gira para trás).
- **Para virar à esquerda:** O motor direito gira para frente e o motor esquerdo para (ou gira para trás).

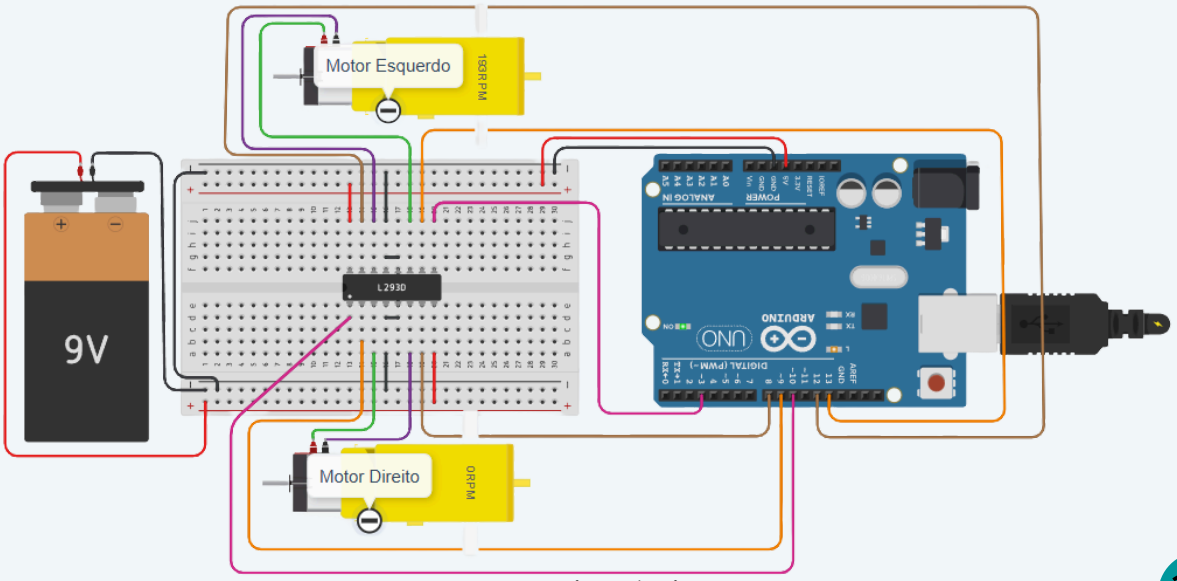
Ajuste na Montagem Virtual:

Para esta missão, precisaremos de dois motores conectados aos dois lados do chip L293D, conforme a Figura 4.5. Considere que o motor Direito montado exatamente como na missão 3.

Configurando o Motor ESQUERDO:

- Conecte os terminais do Motor Esquerdo aos Pinos 3 e 6 (Saída 1 e 2) do L293D, conforme a Figura 5.
- Controle de Direção: Conecte os Pinos 2 e 7 (Entrada 1 e 2) do L293D aos pinos digitais 13 e 12 do Arduino.
- Controle de Velocidade (PWM): Conecte o Pino 1 (Ativar 1 e 2) do L293D ao pino PWM 3 do Arduino.

Figura 4.5 - Circuito com tração dupla e velocidade definida por PWM



Fonte: Autoria própria

A Lógica de Programação (Código C++):

Agora vamos programar o "cérebro" do robô para gerenciar os dois lados simultaneamente. O código ilustrado na Figura 4.6 cria uma sequência clássica de navegação: o robô anda para frente, faz uma curva à direita e, em seguida, uma curva à esquerda.

Figura 4.6 - Exemplo de código C++ para controle de tração dupla

```
  //Motor DIREITO
const int pinoFrenteDireito = 9;
const int pinoTrasDireito = 8;
const int pinoVelocidadeDireito = 10;

  //Motor ESQUERDO
const int pinoFrenteEsquerdo = 13;
const int pinoTrasEsquerdo = 12;
const int pinoVelocidadeEsquerdo = 3;

void setup(){
  //Configura os pinos como saída
 pinMode (pinoFrenteDireito, OUTPUT);
 pinMode (pinoTrasDireito, OUTPUT);
 pinMode (pinoVelocidadeDireito, OUTPUT);
 pinMode (pinoFrenteEsquerdo, OUTPUT);
 pinMode (pinoTrasEsquerdo, OUTPUT);
 pinMode (pinoVelocidadeEsquerdo, OUTPUT);
}

void loop(){
  //Rodar os motores para FRENTE
    digitalWrite(pinoFrenteDireito, HIGH);
    digitalWrite(pinoTrasDireito, LOW);
    digitalWrite(pinoFrenteEsquerdo, HIGH);
    digitalWrite(pinoTrasEsquerdo, LOW);

  //Inicia em linha reta
    analogWrite(pinoVelocidadeDireito, 200);
    analogWrite(pinoVelocidadeEsquerdo, 200);
    delay(2000); //mantém movimento por dois segundos

  //Curva à direita
    analogWrite(pinoVelocidadeDireito, 0);
    analogWrite(pinoVelocidadeEsquerdo, 200);
    delay(2000); //mantém movimento por dois segundos

 //Curva à esquerda
    analogWrite(pinoVelocidadeDireito, 200);
    analogWrite(pinoVelocidadeEsquerdo, 0);
    delay(2000); //mantém movimento por dois segundos
}
```

Fonte: Autoria própria

Entendendo o Movimento

Observe com atenção os blocos de curva no código. Como um robô de duas rodas não possui um "volante" como um carro, a direção é controlada pela diferença de **velocidade** entre as rodas. Quando queremos virar à direita, cortamos a energia do motor direito (analogWrite em 0) e mantemos o esquerdo empurrando. Isso faz o robô usar a roda parada como um eixo de pivô, girando o chassi exatamente para a direção desejada.

MISSÃO 4

O Primeiro Sentido: O Sensor de Linha

Na arena de resgate da OBR, o caminho até a área de desastre é marcado por uma linha preta sobre um fundo branco. Para que o robô consiga seguir essa trilha sozinho, precisamos dar a ele a capacidade de "enxergar" o chão. Fazemos isso utilizando o **Sensor Infravermelho (IR)**, frequentemente chamado de sensor seguidor de linha. (como o módulo TCRT5000).

O princípio de funcionamento baseia-se na reflexão da luz:

- No fundo BRANCO: A luz infravermelha reflete de volta para o sensor. O Arduino recebe um sinal BAIXO (0).
- Na linha PRETA: A cor preta absorve a luz. O Arduino recebe um sinal ALTO (1), entendendo que encontrou a linha.

Ajuste na Montagem Virtual:

Plataformas de simulação como o Tinkercad não possuem um sensor seguidor de linha nativo que "leia" o chão da tela. Para testarmos a nossa lógica de programação, vamos substituir os sensores por **Interruptores Deslizantes**. Dessa forma, você atuará como os "olhos" do robô, mudando a chave para simular quando o sensor tocaria na linha preta (Figura 4.7).

1. Adicione **dois Interruptores Deslizantes** à sua protoboard.
2. **Ligação do Sensor Direito (Interruptor 1):**

- Terminal Esquerdo: Conecte aos 5V (Sinal 1 - Preto).
- Terminal Central: Conecte ao Pino Digital 2 do Arduino.
- Terminal Direito: Conecte ao GND (Sinal 0 - Branco).

3. **Ligação do Sensor Esquerdo (Interruptor 2):**

- Terminal Esquerdo: Conecte aos 5V (Sinal 1 - Preto).
- Terminal Central: Conecte ao Pino Digital 4 do Arduino.
- Terminal Direito: Conecte ao GND (Sinal 0 - Branco).

Direto ao ponto

É obrigatório ligar o pino do GND no interruptor.

Mo mundo real, é possível utilizar o sensor denominado Módulo Seguidor de Linha - **TCRT5000**

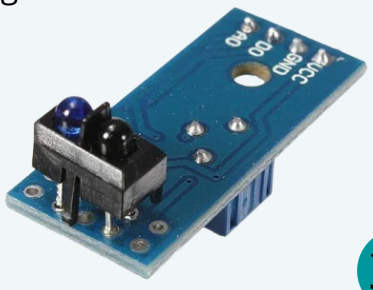
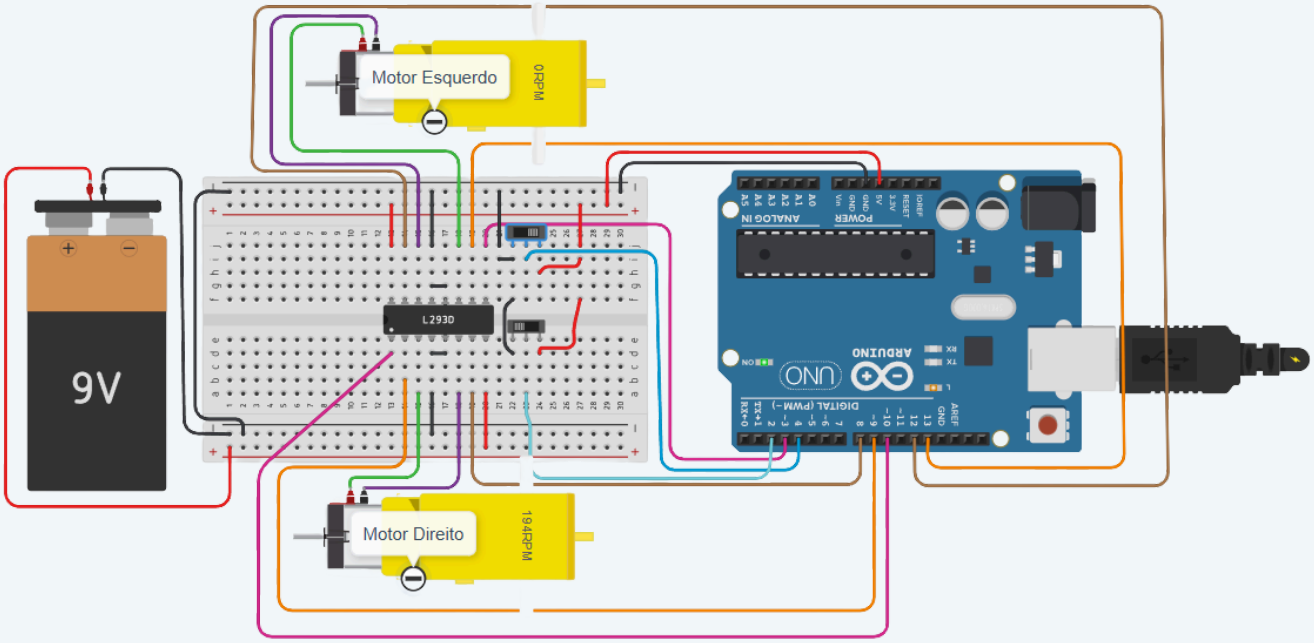


Figura 4.7 - Circuito com sensores simulados por interruptores deslizantes



Fonte: Autoria própria

A Lógica de Programação (O Cérebro Seguidor):

O código agora precisa ler o ambiente constantemente e tomar decisões. O robô fará perguntas simples a si mesmo milhares de vezes por segundo usando as estruturas **if** (se) e **else if** (senão, se), conforme a Figura 4.8.

Ao iniciar a simulação, altere a posição das chaves deslizantes e observe as rotações dos motores na tela. A "dança" de acelerar e frear rodas alternadas é exatamente o que manterá seu robô físico preso à linha na pista de competição!

Figura 4.8 - Exemplo de código para módulo de seguidor de linha

```
  //Motor DIREITO
const int pinoFrenteDireito = 9;
const int pinoTrasDireito = 8;
const int pinoVelocidadeDireito = 10;

  //Motor ESQUERDO
const int pinoFrenteEsquerdo = 13;
const int pinoTrasEsquerdo = 12;
const int pinoVelocidadeEsquerdo = 3;

  //Sensores de Linha
const int sensorDireito = 2;
const int sensorEsquerdo = 4;

void setup(){
  //Configura os pinos como saída
 pinMode (pinoFrenteDireito, OUTPUT);
 pinMode (pinoTrasDireito, OUTPUT);
 pinMode (pinoVelocidadeDireito, OUTPUT);
 pinMode (pinoFrenteEsquerdo, OUTPUT);
 pinMode (pinoTrasEsquerdo, OUTPUT);
 pinMode (pinoVelocidadeEsquerdo, OUTPUT);
 pinMode (sensorDireito, INPUT);
 pinMode (sensorEsquerdo, INPUT);
}

void loop(){
  //Rodar os motores para FRENTE
    digitalWrite(pinoFrenteDireito, HIGH);
    digitalWrite(pinoTrasDireito, LOW);
    digitalWrite(pinoFrenteEsquerdo, HIGH);
    digitalWrite(pinoTrasEsquerdo, LOW);

  //Leitura dos sensores
  // 1 = Preto (linha) | 0 = Branco (piso)
    int leituraDireito = digitalRead(sensorDireito);
    int leituraEsquerdo = digitalRead(sensorEsquerdo);

  // Ambos sensores veem BRANCO (robô centralizado)
  if (leituraEsquerdo == 0 && leituraDireito == 0) {
    //Andar para frente
    analogWrite(pinoVelocidadeDireito, 200);
    analogWrite(pinoVelocidadeEsquerdo, 200);
  }
  //Sensor Esquerdo vê PRETO (robô saindo pela direita)
  else if (leituraEsquerdo == 1 && leituraDireito == 0) {
    //ajustar para a esquerda
    analogWrite(pinoVelocidadeDireito, 200);
    analogWrite(pinoVelocidadeEsquerdo, 0);
  }
   //Sensor Direito vê PRETO (robô saindo pela esquerda)
  else if (leituraEsquerdo == 0 && leituraDireito == 1) {
    //ajustar para a direita
    analogWrite(pinoVelocidadeDireito, 0);
    analogWrite(pinoVelocidadeEsquerdo, 200);
  }
    //Amboes sensores veem PRETO (Linha de chegada)
  else if (leituraEsquerdo == 1 && leituraDireito == 1) {
    //ajustar para a esquerda
    analogWrite(pinoVelocidadeDireito, 0);
    analogWrite(pinoVelocidadeEsquerdo, 0);
  }
}
```

Fonte: Autoria própria

Expandindo as possbilidades de entradas e saídas

4.3

Até o momento, nosso robô simulado aprendeu a se mover usando Motores DC como Saídas e a enxergar o caminho usando sensores de linha simulados como Entradas. No entanto, o microcontrolador Arduino é como um verdadeiro **centro de comando** capaz de gerenciar dezenas de componentes simultaneamente.

Em competições como a OBR, a arena simula um ambiente de desastre. O robô precisará desviar de escombros e identificar vítimas, representadas por bolinhas de resgate, para transportá-las a uma área segura. Para que o robô tome essas decisões, precisamos apresentar a ele novos **sentidos** e **membros**.

Novos Componentes de Entrada (Sensores):

São os componentes que captam informações do mundo externo e enviam para o Arduino.

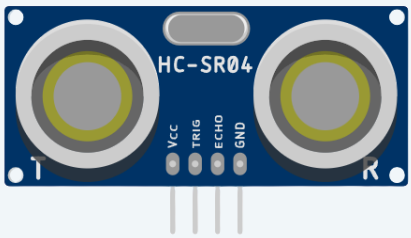
- **Sensor de Distância (Ultrassônico HC-SR04):**

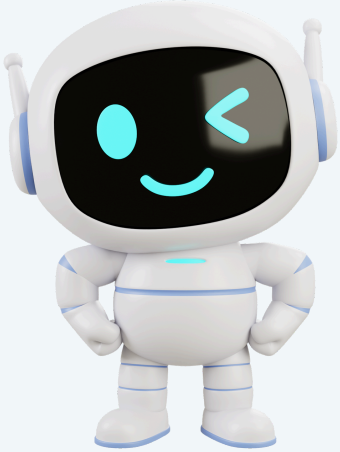
Na pista da Olimpíada Brasileira de Robótica (OBR), o caminho nem sempre estará livre. Em determinados momentos, o robô encontrará obstáculos (como blocos de isopor ou madeira) bloqueando a fita preta. Ele precisa ser capaz de detectar esse obstáculo de longe, parar, desviar e encontrar a linha novamente mais à frente.

Para dar ao nosso robô esse **sentido de profundidade**, utilizamos o Sensor Ultrassônico HC-SR04.

Direto ao ponto

A prática do uso do sensor ultrassônico será vista nos capítulos subsequentes





Saiba mais sobre o sensor de distância HC-SR04

https://www.youtube.com/watch?v=TX62dkfrb34

YouTube

Como o robô mede a distância?

O sensor ultrassônico funciona exatamente como a ecolocalização de um morcego ou o sonar de um submarino:

1. O pino **Trigger (Gatilho)** emite um pulso sonoro de alta frequência (inaudível para nós).
2. Esse som viaja pelo ar e bate no obstáculo.
3. O som reflete (eco) e volta para a placa, sendo captado pelo pino Echo (Eco).

Sabendo a velocidade do som no ar, o **cérebro** do robô consegue calcular o tempo exato que o som demorou para ir e voltar, descobrindo a distância do objeto em centímetros.

Do Virtual para o Mundo Real (O Uso de Bibliotecas):

Na programação profissional e em competições de robótica, nós não escrevemos do zero toda a matemática complexa para calcular a velocidade do som. Em vez disso, utilizamos Bibliotecas (como a famosa **NewPing.h**). Uma biblioteca é um pacote de códigos prontos que simplifica o uso de um componente complexo, permitindo que você descubra a distância de um obstáculo usando apenas uma linha de código.

Como os simuladores *online* possuem restrições técnicas para a importação e instalação de bibliotecas externas, tentar simular esse sensor agora exigiria escrever um código manual muito diferente daquele que você realmente usará no seu robô físico.

Sensores de Cor:

As vítimas do resgate são divididas por cores (ex: bolinhas prateadas para vítimas vivas e pretas para vítimas mortas), e o robô precisa saber qual está pegando para deixá-la no local correto. No mundo real, usamos módulos como o **TCS3200**. No Tinkercad, como não temos um sensor de cor nativo, treinamos essa lógica de programação utilizando **Fotoresistores (LDR)** para ler variações de luminosidade, ou simplesmente usamos a nossa técnica de Interruptores Deslizantes para simular a detecção da cor e validar o código.

Direto ao ponto

Na prova da OBR, o sensor **TCS3200** é utilizado para identificar a cor das bolinhas na área de resgate, permitindo que o robô diferencie as vítimas vivas (prateadas) das mortas (pretas) para transportá-las ao local correto de evacuação.

Síntese do Capítulo:

Chegamos ao fim da nossa jornada pelo ambiente do Tinkercad! Até aqui, utilizamos a simulação de circuitos não apenas para evitar a queima de componentes reais, mas para construir a fundação lógica e elétrica de um robô de resgate.

Neste capítulo, você deixou de ser apenas um montador de peças e passou a ser o **programador do cérebro da máquina**. Vamos relembrar as grandes conquistas das nossas missões:

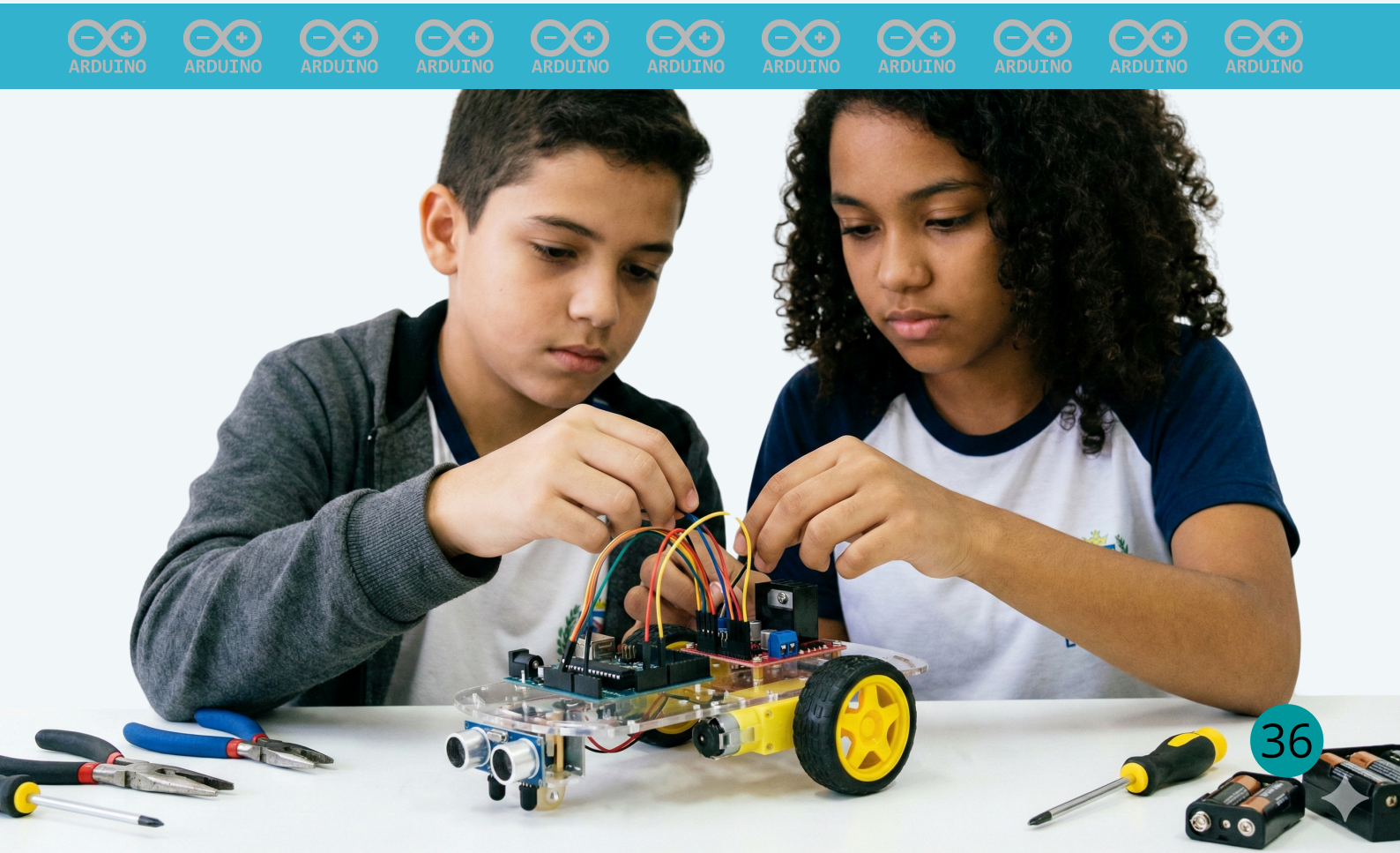
- **Dominamos a Eletrônica de Potência:** Aprendemos que o Arduino é frágil e não pode alimentar motores diretamente. Usamos a Ponte **H (L293D)** como nosso "escudo e músculo", separando a energia inteligente (5V) da energia bruta (Bateria de 9V).
- **Controlamos a Direção e a Velocidade:** Entendemos a diferença entre engatar a marcha usando sinais digitais (digitalWrite) para ditar o sentido da corrente, e pisar no acelerador usando o PWM (analogWrite) para ter um controle fino e não derrapar na pista.
- **Fizemos o Robô Manobrar:** Aplicamos o conceito de Tração Diferencial. Ao controlar dois motores de forma independente, provamos que é possível fazer curvas precisas parando uma roda e acelerando a outra, dispensando a necessidade de um sistema de direção complexo como o de um carro.
- **Demos o Primeiro Passo para a Autonomia:** Na nossa última missão, simulamos a visão do robô. Usando interruptores para representar os Sensores de Linha, criamos uma lógica de perguntas e respostas (**if** e **else if**) que permite ao Arduino tomar decisões sozinho, corrigindo a própria rota para se manter no caminho.

O Próximo Nível: O Tinkercad foi incrível para entendermos como os fios se conectam e como a eletricidade flui. No entanto, um robô da Olimpíada Brasileira de Robótica (OBR) não vive em uma protoboard parada; ele vive em uma arena 3D cheia de física, atrito, rampas e falhas na linha.

Para testarmos nossa lógica de programação em um ambiente de competição real, precisamos de um simulador que nos dê uma pista completa e um robô já montado. É exatamente isso que faremos no próximo capítulo ao entrarmos no universo do **sBotics.**

5 Do Virtual para o Real

Este capítulo marca a transição do laboratório virtual para a bancada de montagem física do robô. O objetivo não é focar em projetos mecânicos complexos, mas guiar alunos e profes[illegible]s na integração segura de componentes reais, superando desafios práticos que a simulação não prevê, como atrito, peso e calibração de luz, para garantir a construção de uma máquina robusta e competitiva para a OBR.



A Base Mecânica: Além de um Kit Comercial

5.1

No mundo virtual do Tinkercad, arrastávamos componentes para o ecrã e tudo encaixava na perfeição, a flutuar no espaço. No mundo real, a gravidade, o atrito e o peso do material entram em cena. O primeiro passo da nossa montagem física é construir o "esqueleto" e os "músculos" do robô: o **chassis** e os **motores**.

ATENÇÃO: O uso de robôs móveis pré-fabricados leva à desclassificação imediata da equipe, pois a OBR exige que a construção e o planeamento do robô sejam fruto do trabalho e criatividade da equipe competidora.

Inovar para vencer

Em vez de se limitarem a aparafusar peças de um kit genérico, este é o momento de pensarem "fora da caixa". A OBR valoriza imenso a inovação através do **Prêmio Maker**, atribuído às equipas que criam as suas próprias partes mecânicas em vez de usarem blocos padrão disponíveis para compra.

Ideias para o Chassis Personalizado:

MDF ou Madeira: São materiais fáceis de cortar, furar e colar. Permitem criar robôs extremamente robustos e com facilidade de adaptação.

Acrílico ou Policarbonato: Excelentes opções para um aspeto limpo e profissional.

Impressão 3D: O limite é a imaginação! Permite desenhar suportes à medida para os sensores e encaixes perfeitos para os motores.

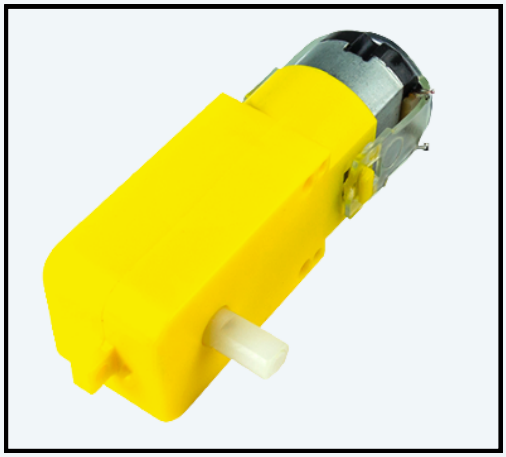
Materiais Reciclados: Capas de CD antigas, tubos de PVC ou peças de brinquedos velhos podem ser transformados em um chassis incrível, leve e sustentável.

Independentemente do material que escolherem para o chassi, os motores mais utilizados para a base rolante são os motoredutores (os famosos motores "amarelinhos" com caixa de redução). O sistema de engrenagens interno reduz a velocidade excessiva do motor e multiplica a sua força. É isso que garante que o robô consiga carregar o peso das baterias de lítio e consiga subir as rampas da arena sem bloquear.

Pit Stop

DICA

Antes de fixarem os motores no vosso chassis criativo, soldem os fios (jumpers) diretamente nos contactos de cobre do motor. Tentar soldar cabos com o motor já embutido no chassis é bastante frustrante e resulta, quase sempre, em maus contactos a meio da prova!

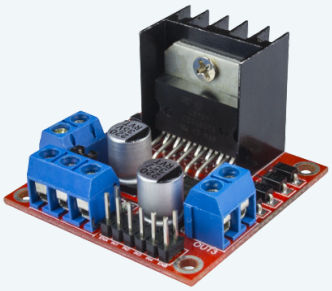


Motor com caixa de redução 5V

A Ponte H e Baterias 5.2

No Tinkercad, o circuito foi alimentado por uma pilha de 9V infinita e utilizava o pequeno chip L293D. Na vida real, um robô com esta configuração provavelmente vai andar alguns centímetros e reiniciar. A arena da OBR exige força, atrito e muita energia. É por isso que precisamos de dar um "upgrade" ao coração e aos músculos do robô.

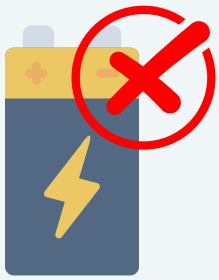
Para substituir o frágil chip L293D, a maioria das equipas utiliza o Módulo **L298N**, que suporta correntes elétricas muito maiores sem queimar e dissipa o calor de forma eficiente.



A Ilusão da Pilha de 9V

A pilha quadrada de 9V armazena carga insuficiente para alimentar os motordutores por muito tempo. Estes compoentes demandam muita energia quando tentam arrancar ou subir a rampa de resgate.

Se usar uma pilha de 9V comum, a tensão vai cair drasticamente assim que os motores exigirem força, fazendo com que o Arduino se desligue e reinicie sozinho no meio da prova. A solução definitiva é utilizar Baterias de Lítio.

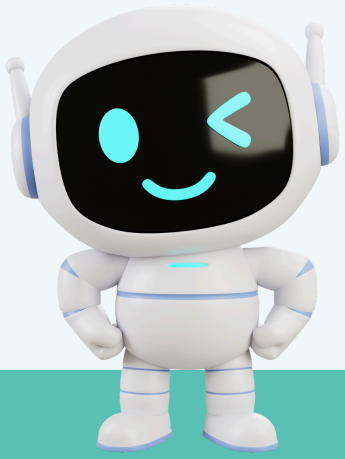


O Grande Dilema: Uma ou Duas Fontes de Energia?
Quando o robô sobe uma rampa, os motores exigem um pico extremo de força. Se a bateria não for robusta o suficiente, ocorre uma queda brusca de tensão no circuito. Se o Arduino estiver ligado a essa mesma bateria, ele entende que a energia acabou e reinicia sozinho. O seu robô perde toda a lógica naquele instante e fica inoperante.

A arquitetura mais segura e utilizada pelas equipes de alto nível é o Sistema de **Duas Fontes.** Você separa completamente a energia do "cérebro" da energia dos "músculos". Como os circuitos estão isolados, os motores podem puxar a força que quiserem sem jamais roubar a estabilidade do Arduino.

As Combinações de Energia Recomendadas:
As baterias quadradas de 9V não são sucientes para suportar a demanda de energia durante a competição. O seu robô precisa de baterias recarregáveis de alta descarga. Aqui estão as duas configurações ideais para a pista:

Veja mais opções sobre a alimentação do robô



Vídeo: Programando no Arduino com C++

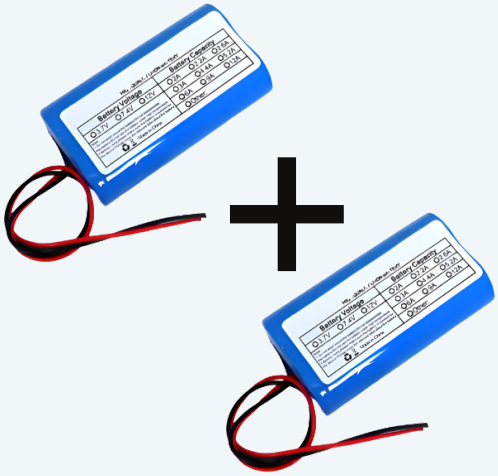
YouTube

Alimentação do robô

OPÇÃO 1

O Padrão de Segurança
(18650 + 18650)

Esta é a configuração com o melhor custo-benefício e excelente para a rotina de testes. Você utiliza um suporte com baterias Li-Ion 18650 ligado exclusivamente ao Arduino (através do plug P4 ou pino Vin), e um segundo suporte independente, também com baterias 18650, ligado exclusivamente à Ponte H (L298N) para os motores. Padroniza-se o uso das baterias, facilita-se a recarga e garante-se muita autonomia de forma segura.

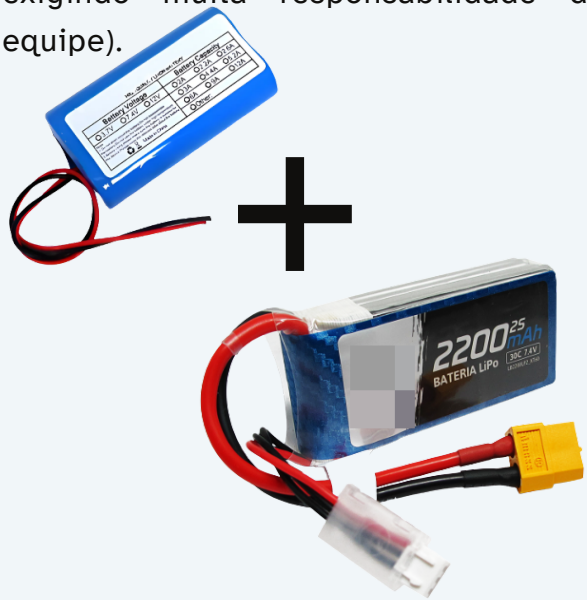


Bateria de Lition 18650 7,4V 2.400mAh

OPÇÃO 2

Opção 2: Alto Desempenho
(18650 + LiPo)

Se o seu robô for mais pesado ou precisar de uma explosão de velocidade extra para vencer os obstáculos, você mantém a bateria 18650 alimentando o Arduino com estabilidade, mas utiliza uma bateria de drone **LiPo** (Polímero de Lítio) dedicada apenas aos motores na Ponte H. A LiPo entrega energia instantânea com uma taxa de descarga bem maior. (Aviso: Baterias LiPo exigem carregadores específicos e nunca podem descarregar totalmente, exigindo muita responsabilidade da equipe).



Bateria LiPo 7,4V 2.200mAH

Boas Práticas de Cabeamento

5.3

No Tinkercad, criar uma conexão elétrica é fácil: você clica, arrasta e uma linha reta e colorida aparece flutuando perfeitamente na tela. Na vida real, se você apenas conectar os fios de qualquer jeito, o seu robô vai se transformar rapidamente no temido "ninho de rato".

Na Olimpíada Brasileira de Robótica (OBR), a organização dos fios não é apenas uma questão de estética; é uma questão de sobrevivência na pista. Fios soltos podem enroscar nas rodas, bloquear a visão dos sensores ou se soltar durante um impacto, custando pontos preciosos.

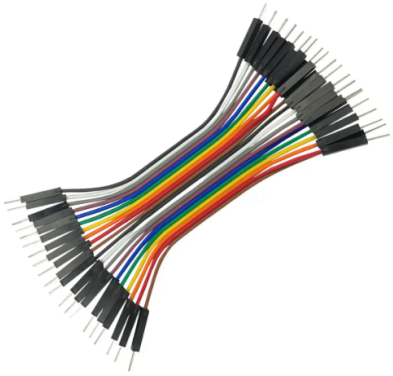
O Arsenal de Conexão: Cabos Jumper

Para ligar o Arduino aos módulos (como a Ponte H e os sensores), utilizamos os Cabos Jumper. Eles vêm colados em "fitas" coloridas e possuem três tipos principais, definidos pelas suas pontas:

Macho-Macho (M-M): Pinos de metal nas duas pontas. (Ideais para ligar o Arduino à protoboard).

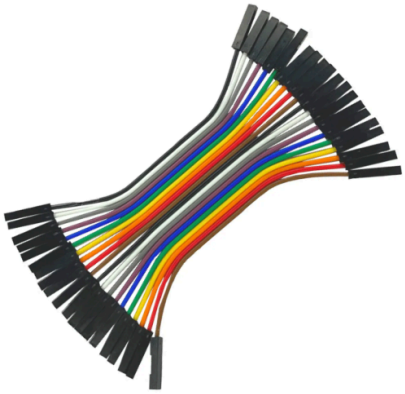
Macho-Fêmea (M-F): Um pino de metal de um lado e um "furo" do outro. (Os mais usados para ligar sensores ao Arduino).

Fêmea-Fêmea (F-F): Possui “furos” nas duas pontas.



Cabo M-M

Cabo M-F



Cabo F-F

Padronização de cores:

Para evitar confusão, é interessante manter o mesmo padrão de cores utilizado no simulador para os terminais positivo (+5V) e negativo (Vin) dos circuitos:

Fio vermelho: +5V

Fio preto: GND

Organizando os cabos:

Um robô fácil de entender visualmente é um robô fácil de consertar. Aplique estas boas práticas de montagem:

- **Trance os fios de um mesmo componente:** O sensor de linha possui três fios (VCC, GND e Sinal). Em vez de deixá-los soltos, descole um trio de jumpers da fita mantendo-os unidos, ou trance-os levemente como uma corda. Isso cria "cabos únicos" e limpa o visual do chassi.

- **Use Abraçadeiras de Nylon:** Elas são as melhores amigas do competidor. Agrupe os fios que passam pelo mesmo caminho e prenda-os ao chassi.

- **Cuidado com as partes móveis:** Certifique-se absolutamente de que nenhum fio passa perto das engrenagens dos motores amarelinhos ou das rodas. O atrito constante vai desencapar o fio e pode causar um curto-circuito fatal para o Arduino.

Foco OBR

Um projeto bem-sucedido não é aquele que tem mais componentes, mas aquele que **falha menos**. Gastar 10 minutos a mais organizando os jumpers com fita isolante ou abraçadeiras pode evitar horas de dor de cabeça procurando um mau contato invisível antes do robô entrar na arena.

Posicionamento e Calibração do Sensor TCRT5000

5.4

No ambiente de simulação, a iluminação é sempre perfeita, não existem sombras e o chão é perfeitamente liso. Na vida real, a arena da OBR tem reflexos, a luz do ginásio pode mudar ao longo do dia e o chão de MDF pode ter pequenas irregularidades.

Para que o robô consiga **identificar** a linha preta com clareza no mundo físico, precisamos de dar especial atenção à montagem e calibração dos Módulos Sensores de Linha (TCRT-5000).

1. **Buscando a altura ideal:**

Um erro clássico na montagem mecânica é fixar o sensor de qualquer forma. O sensor infravermelho funciona emitindo luz e lendo o reflexo.

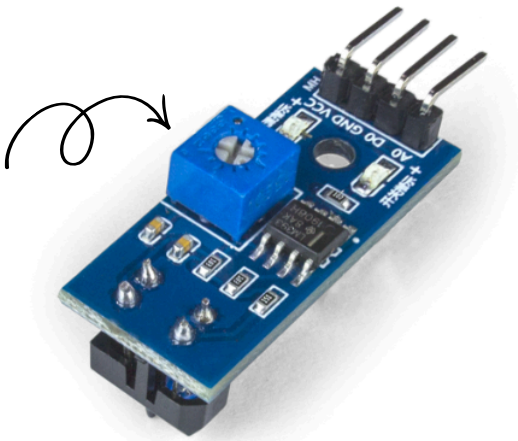
- Se o sensor ficar muito colado ao chão (a menos de 3 milímetros), ele pode raspar nas irregularidades da pista, prender-se nos redutores de velocidade (obstáculos da OBR) ou até mesmo ser arrancado na subida da rampa.

- Se ficar muito alto (a mais de 3 centímetros), a luz infravermelha dispersa-se no ar e o sensor deixa de conseguir distinguir o que é branco e o que é preto.

- **A Regra de Ouro:** Fixe o módulo no chassis de forma a que os pequenos LEDs (o emissor transparente e o recetor escuro) fiquem suspensos a uma distância de 1 a 1,5 centímetros do chão da arena.

2. **A Calibração do sensor**

O módulo Sensor Óptico Reflexivo TCRT-5000 possui um **potenciômetro**, uma pequena peça azul com um parafuso metálico no topo, cuja função é ajustar a sensibilidade de detecção de luz, processo chamado de calibração.



Módulo Sensor Óptico Reflexivo TCRT-5000

Passo a Passo para a Calibração:

1. **Posicionamento:** Coloque o robô ligado na pista de testes. Posicione o sensor esquerdo exatamente sobre o fundo BRANCO.
2. **O Ajuste Fino:** Com uma pequena chave de estrela (Philips) ou de fendas de precisão, rode o parafuso do potenciómetro muito lentamente. Verá que a placa tem dois pequenos LEDs SMD (luzes na própria placa verde). Um indica que o módulo tem energia, o outro acende ou apaga dependendo do que o sensor "vê".
3. **O Limiar:** Gire o parafuso até encontrar o ponto exato onde o LED indicador acende quando está sobre o branco e apaga imediatamente quando move o robô para cima da linha PRETA.
4. **Repita o Processo:** Faça exatamente o mesmo ajuste para os outros sensores do robô.

Testes dos sensores:

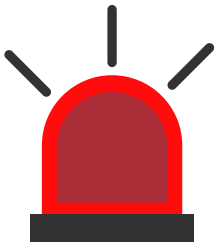
Antes de colocar o robô a andar sozinho, faça o teste de bancada: levante o robô e passe o dedo por baixo dos sensores. Se o LED da placa reagir à aproximação do seu dedo, a calibração está no caminho certo. A partir deste momento, o Arduino receberá finalmente os sinais perfeitos de 0 (LOW) e 1 (HIGH) que programamos no simulador.

O Teste de Bancada e o Primeiro Upload

5.5

O robô tem um chassis robusto, uma fonte de energia potente separada do processador central (Arduino) e os sensores infravermelhos perfeitamente calibrados para a luz ambiente. Agora, só falta transferir a lógica programada no Tinkercad para o microcontrolador Arduino físico.

protocolo de SEGURANÇA



NUNCA realizar o primeiro upload com as rodas a tocar na mesa de testes

Assim que o Arduino receber o código e reiniciar, ele vai executar as instruções imediatamente. Se o código mandar andar para a frente a toda a velocidade, o vosso robô vai saltar da mesa e espatifar-se no chão em frações de segundo!

A transferência do programa necessita da instalação do software Arduino IDE no seu computador. Atualmente o software está na versão 2.3.7 (fevereiro de 2026).

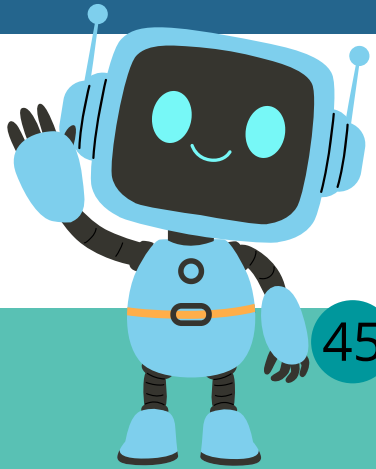
DICA: Você pode configurar a versão do Arduino IDE para **Português do Brasil**.

Arduino IDE 2.3.7

Release notes

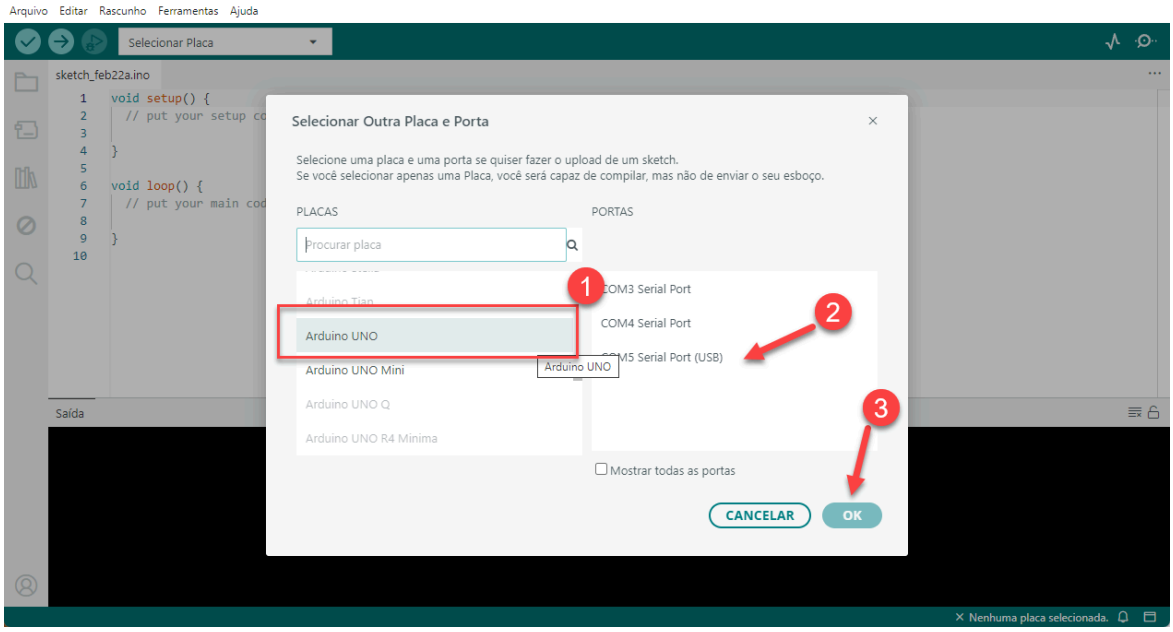
O programa Arduino IDE pode ser baixado em:

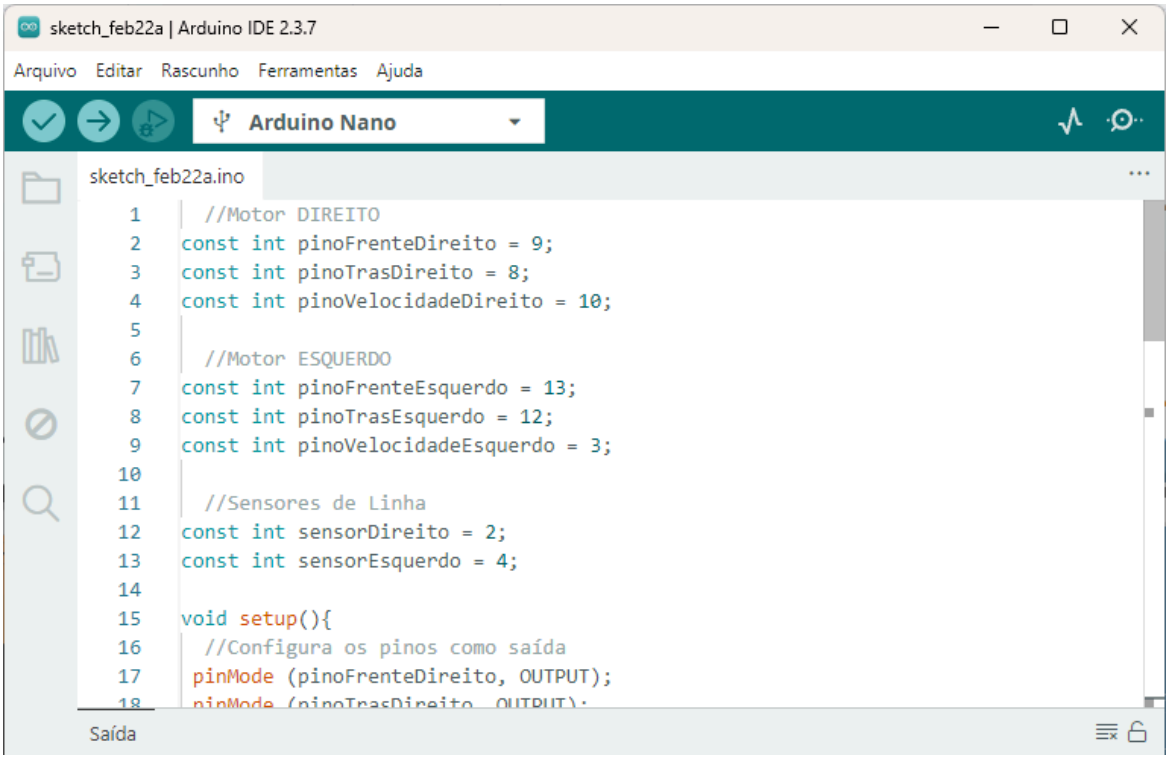
https://www.arduino.cc/en/software/#ide



Após a instalação, abra o software oficial do Arduino (IDE) no computador e cole o mesmo código realziado na Missão 4 do Capítullo anterior (Tinkercad).

1. Conecte o cabo USB ao Arduino e ao computador.
2. Clique no menu *Ferramentas* > *Placa* e selecionem **Arduino Uno**.
3. Clique no menu *Ferramentas* > *Porta* e selecione a porta COM correspondente ao robô (geralmente é a que tem o número mais alto ou que sinalizada para "Arduino").
4. Clique no botão redondo com uma seta a apontar para a direita **Carregar**.





Erros de iniciante e Resolução de Problemas 5.6

É muito comum que o robô não funcione perfeitamente no primeiro clique. Se algo der errado, não apague o seu código! Verifique esta lista dos 5 problemas mais comuns nas montagens de iniciantes:

Erro 1

O Motor Invertido (A Roda Gira para Trás)

O código diz para ir para a frente, mas uma das rodas (ou ambas) gira para trás. O instinto é ir ao código e trocar os comandos HIGH e LOW. Não faça isso! O problema é apenas físico.

Foco OBR

Solução: Com a sua chave Philips, desaperte os dois fios desse motor no borne da Ponte H (L298N). Inverta a posição dos dois fios (o da esquerda vai para a direita e vice-versa) e volte a apertar. A roda passará a girar para o lado certo e o seu código continua limpo.

Erro 2

A Porta COM Não Aparece (O Cabo USB "Falso")

Você liga o Arduino ao computador, as luzes acendem, mas o IDE não encontra nenhuma Porta COM para fazer o *upload*.

Foco OBR

Solução: Muitos alunos utilizam cabos USB de carregadores de telemóvel antigos. O problema é que a maioria desses cabos são "Apenas Carga" e não possuem os fios internos de transmissão de dados. Troque por um cabo USB que tenha a certeza que transmite dados e a Porta COM aparecerá.

Erro 3

O Código Carrega, Mas o Robô Não Anda

O IDE diz "Transferência Concluída", o robô está ligado à bateria, mas os motores não fazem nenhum som e nem se mexem.

Foco OBR

Solução: Verifique o Terra Comum. Se estiver usando uma bateria para o Arduino e outra para a Ponte H, é obrigatório existir um fio jumper a ligar o pino **GND** do Arduino ao borne **GND** da **Ponte H L298N**. Sem esta ligação, o circuito de comunicação não se fecha e a Ponte H não funciona.

Erro 4

O pino do Arduino trocado

O robô anda para a frente, mas quando o sensor esquerdo vê a linha preta, é a roda direita que reage de forma estranha, ou o robô ignora a linha completamente.

Foco OBR

Solução: Revise o cabeamento batendo com o código. No Tinkercad, você definiu que o *sensorDireito* era o pino 2. Na pressa da montagem física, é muito comum espetar o jumper no pino 3 ou 4. Verifique fio por fio se a montagem física espelha rigorosamente o que está escrito no cabeçalho do seu código.

Erro 5

O “Fantasma” do mau contato

O robô funciona perfeitamente quando está com as rodas no ar, mas assim que o coloca no chão para andar, ele desliga, reinicia ou um motor falha.

Foco OBR

Solução: O atrito com o chão gera vibração. Jumpers (fios) de má qualidade ou oxidados podem afrouxar com essa vibração. Substitua os cabos suspeitos e utilize fita isoladora ou abraçadeiras de nylon para prender os chicotes de fios ao chassi, impedindo que balancem durante o movimento.

Conclusão

Se você chegou até aqui e o seu robô está acompanhando a linha preta no chão da sala, **parabéns!** Você acaba de dar o passo mais importante e desafiador no mundo da robótica autônoma.

Ao longo deste guia, você deixou de ser apenas um usuário de tecnologia para se tornar um **criador**. Você compreendeu como a eletricidade flui através da Ponte H, aprendeu a programar o "cérebro" do Arduino com lógica e matemática, e, o mais importante, descobriu como contornar os desafios do mundo real na hora de calibrar os sensores e organizar os cabos do seu chassi.

O robô que está agora na sua bancada é a sua porta de entrada para a **Olimpíada Brasileira de Robótica (OBR)**. Lembre-se de que a engenharia é feita de testes, erros e melhorias contínuas. O seu robô vai sair da pista algumas vezes, a luz do ginásio vai exigir novas calibrações e as baterias vão acabar na pior hora possível. Tudo isso faz parte do jogo dos **campeões**.

Este manual entregou a você a base sólida. Daqui para frente, a arena é sua. Explore novos sensores, desenhe peças inovadoras em 3D, estude formas de resgatar as vítimas e nunca deixe de compartilhar o conhecimento com a sua equipe.

Nos vemos na pista.
Boa sorte e bons códigos!

Referências

ARDUINO. Software (IDE). Versão 2.3.7. Disponível em: https://www.arduino.cc/en/software/#ide. Acesso em: 1 mar. 2026.

AUTODESK. Tinkercad. Disponível em: https://www.tinkercad.com/. Acesso em: 1 mar. 2026.

BRINCANDO COM IDEIAS. Sensor de Distância! Conheça o Módulo HC-SR04 - DESAFIO MAKER. YouTube, 27 out. 2021. Disponível em: https://www.youtube.com/watch?v=TX62dkfrb34. Acesso em: 1 mar. 2026.

BRINCANDO COM IDEIAS. Pilhas e Baterias para Robôs com Arduino #MaratonaMaker. YouTube, 31 ago. 2022. Disponível em: https://www.youtube.com/watch?v=AFXEHUU6TPg. Acesso em: 1 mar. 2026.

LABORATÓRIO DE CIRCUITOS. O ÚNICO vídeo que você precisa ASSISTIR para iniciar no ARDUINO. YouTube, 2 fev. 2026. Disponível em: https://www.youtube.com/watch?v=XYs0lm7Pe9M. Acesso em: 1 mar. 2026.

MANUAL DO MUNDO. Use um Arduino sem ter Arduino! #ManualMaker Aula 5, Vídeo 1. YouTube, 21 mar. 2019. Disponível em: https://www.youtube.com/watch?v=CrHJj4OQ6Sw. Acesso em: 1 mar. 2026.

OLIMPÍADA BRASILEIRA DE ROBÓTICA. Manual da OBR 2025. Disponível em: https://obr.robocup.org.br/documentos-e-manuais/. Acesso em: 1 mar. 2026.

OLIMPÍADA BRASILEIRA DE ROBÓTICA. Modalidade Prática. Disponível em: https://obr.robocup.org.br/modalidade-pratica/. Acesso em: 1 mar. 2026.

Sobre o autor e o Projeto

Jonas Fernandes da Silva é formado em Telecomunicações e especialista em Docência para a Educação Profissional e Tecnológica, ambas pelo Instituto Federal da Paraíba (IFPB). É mestre em Informática pela Universidade Federal da Paraíba (UFPB) e concluinte da especialização em Robótica Educacional pela Universidade Federal Rural de Pernambuco (UFRPE). Atualmente, é professor no IFPB Campus Patos e um grande entusiasta da Robótica Educacional. Em sua vivência diária no ensino técnico e superior, acompanha de perto o desenvolvimento de estudantes em áreas fundamentais da tecnologia, como Redes de Computadores, Sistemas Digitais e Eletrônica. Trabalhando lado a lado com alunos ingressantes no ensino médio, percebeu as verdadeiras barreiras que as equipes iniciantes enfrentam ao tentar tirar os seus primeiros robôs do papel.

Este guia nasceu da necessidade de criar uma ponte direta e sem rodeios entre a teoria da sala de aula e a prática da arena. O objetivo deste projeto é democratizar o acesso à Olimpíada Brasileira de Robótica (OBR), entregando a professores e alunos, principalmente de escolas públicas, um método ágil (Lean) para construir, programar e testar robôs de resgate, minimizando as frustrações e focando no aprendizado real.

Acredito que a robótica não é apenas sobre montar peças, mas sobre desenvolver a capacidade de resolver problemas complexos com criatividade e trabalho em equipe.

Vamos nos conectar?

Caso este guia tenha ajudado a sua equipe, ou se quiser trocar ideias sobre metodologias de ensino e robótica, sinta-se à vontade para entrar em contato:

E-mail: fernandes.jonas@ifpb.edu.br

Currículo Lattes: http://lattes.cnpq.br/0924424581894796

Instagram: jonas.f2